Chronos: sistema de simulación de horarios

Por
Javier García Lorenzo
Carlos Lozano Casado

UNIVERSIDAD COMPLUTENSE
MADRID

Grado en Ingeniería Informática
Facultad de Informática

Javier García Lorenzo
Carlos Lozano Casado
Chronos: sistema de simulación de horarios

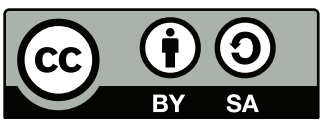
Madrid, 2018–2019

Autorización de difusión y uso

Se autoriza a la UCM (Universidad Complutense de Madrid) a difundir y usar el trabajo realizado con fines académicos y mencionado a los autores, tanto la memoria como el código.

Javier García Lorenzo
Carlos Lozano Casado

Madrid, Mayo 2019



Agradecimientos

A mi familia. Por enseñarme que hay que esforzarse por cumplir nuestros sueños, que hay que ser constante en el proceso y que se le puede dar la vuelta a cualquier situación. También a Pascal. Por ayudarme, por animarme en los momentos de flaqueza, por estar cuando se le necesita, por todo.
Javier

A mi compañero de TFG, Daster, por permitirme ser parte de este proyecto, y si pedirle perdón por cualquier inconveniencia que le haya causado.
A la asociación ASCII, no me puedo imaginar mi paso por la universidad sin este lugar y sin las personas que lo componen.
Carlos

Sobre TEFLON

Teflon(cc0 1.0(documentación) MIT(código))es una plantilla de LaTeX creada por David Pacios Izquierdo con fecha de Enero de 2018. Con atribuciones de uso CC0.

Esta plantilla fue desarrollada para facilitar la creación de documentación profesional para Trabajos de Fin de Grado o Trabajos de Fin de Máster. La versión usada es la 2.1.

V:2.1 Overleaf V2 with XeLaTeX, margin 1in, bib

Contacto
Autor: David Pacios Izquiero
Correo: dpacios@ucm.es
ASCII: ascii@ucm.es
Despacho 110 - Facultad de Informática

Índice general

Resumen

Con nuestro trabajo queremos facilitar la organización del horario para profesores y alumnos. Pretendemos facilitar esa organización con nuestra aplicación. Nuestra aplicación puede organizar el horario seleccionando las asignaturas que quiere cursar y le indica si están o no están disponibles.
Hemos estado viendo aplicaciones similares a la que pretendíamos crear, pero no todas hacían lo que nosotros pretendíamos que hicieran. Pero nos dieron una idea de lo que queríamos hacer con la nuestra. En este punto pretendíamos tener una aplicación con una interfaz sencilla para el usuario y sin coste alguno.
Para ello, programamos la interfaz de la aplicación mediante CSS y Javascript, y nos coordinamos en el equipo mediante la metodología Kanban.

Palabras Clave:CSS, Javascript, jQuery, horario, interfaz, estudiantes, docentes y herramientas.

Abstract

With our work we want to facilitate the organization of the schedule for teachers and students. We intend to facilitate that organization with our application. Our application can organize the schedule by selecting the subjects you want to attend and it tells you if they are available or not.
We have been seeing applications similar to the one we wanted to create, but not all of them did what we wanted them to do. But they gave us an idea of what we wanted to do with ours. At this point we pretended to have an application with a simple interface for the user and without any cost.
To do this, we program the interface of the application through CSS and Javascript, and we coordinate on the team using the Kanban methodology.

Key Words:CSS, Javascript, jQuery, schedule, interface, students, teachers and tools.

Capítulo 1

Introducción

En este capítulo se tratará:

1.1. Motivación para realizar el proyecto

Durante los años que hemos estado estudiando en la Complutense hemos realizado grandes malabares ofimáticos para poder cuadrar los horarios del año siguiente, o bien porque al quedarnos algo para Septiembre escogíamos en fechas muy malas o porque los créditos no daban para coger ciertas asignaturas. Es muy común ver chats con una gran cantidad de alumnos, que llegada la fecha de matriculación comienzan a pasarse tablas en Excel, Words o incluso en papel para indicar los horarios que quieren. Estas fechas son de bastante nerviosismo entre el alumnado, puestos que es difícil siempre cuadrar tu horario y observas una gran cantidad de gente que se matricula antes que tú y a los cuales preguntas cuántas plazas quedan para volver a ajustar tu calendario manual. El año pasado como último año de carrera decidimos tomar medidas.
Hemos propuesto la creación de un sistema simulador de matrículas con el objetivo de poder ayudar a todas esas personas, para ello lo primero que se hizo fue investigar si existía alguna herramienta que realizara algo parecido, y de ser así, cómo se usa. Este proceso de documentación nos llevó a ver que el único sistema que se parecía era el de ucmnet, pero no tenía todas las funcionalidades ni te permitía simular un calendario sin que sea la fecha de matriculación. Por lo tanto, nos encontrábamos ante un vacío que perfectamente podíamos cubrir.

1.2. Objetivos a realizar

- Crear un sistema simulador de horarios.
- Implementarlo vía web.
- Compatibilidad con sistemas móviles.
- Adaptado para visibilidad reducida.
- Que cumpla un estándar para que sea escalable.

1.3. Estructura del documento

El trabajo ha sido desarrollado a lo largo de siete capítulos más uno de bibliografía, que contiene los enlaces de referencia que se han utilizado para documentar las distintas tecnologías usadas. Estos enlaces de referencia se mostrarán a través de hipervínculos a los distintos sitios. En los capítulos se profundizará los conceptos marcados en el resumen del siguiente índice de contenidos:

- **Capítulo 1**. Introducción. Marca el comienzo del trabajo y la motivación del mismo.
- **Capítulo 2**. Estado del arte. Se habla de las herramientas que se han utilizado, de las aplicaciones similares en las que nos hemos inspirado y las tecnologías usadas.
- **Capítulo 3**. Metodología y tecnología. Vamos a utilizar la metodología Kanban para mostrar las distintas fases del proyecto desde su inicio. También vamos a profundizar en las tecnologías usadas.

- **Capítulo 4**. Diseño de de la aplicación. Muestra final de la aplicación, explicación de la interfaz, gestión de riesgos y diseño de la misma.
- **Capítulo 5**. Arquitectura e implementación. Se van a mostrar las distintas capas de la aplicación.
- **Capítulo 6**. Resultados. Discusión sobre los resultados finales del proyecto.
- **Capítulo 7**. Conclusiones y trabajo futuro. Conclusiones y trabajo futuro.
- **Capítulo 8**. Bibliografía.

Los capítulos uno y siete se pueden encontrar también traducidos al inglés como se específica en la normativa de los Trabajos de Fin de Grado de la UCM.

Chapter 1

Introduction

In this chapter we will talk about:

1.1. Motivation

During the years that we have been studying in the Complutense, we have done great office juggling to fit the schedules of the following year, or because when we stayed for September we chose very bad dates or because the credits did not allow us to take certain subjects. It is very common to see chats with a large number of students, when the registration date begins to pass tables in Excel, Words or even on paper to indicate the times they want. These dates are quite nervous among students, positions that it is difficult to always square your schedule and observe a large number of people who enroll before you and to which questions how many places are left to re-adjust your manual calendar. Last year as the last year of the race we decided to take action.
We have proposed the creation of a license plate simulator system with the aim of being able to help all those people, for this the first thing that was done was to investigate if there was a tool that could do something similar, and if so, how it is used. This documentation process led us to see that the only system that looked like it was the ucmnet, but it did not have all the functionalities nor did it allow you to simulate a calendar without it being the registration date. Therefore, we were facing a vacuum that we could perfectly cover.

1.2 Objetives

- To create a simulator system of schedules.
- Implement it vía web.
- Compatibility with moblie systems.
- Adapted for low vision.
- That meets a standard for it to be scalable.

1.3. Structure of dissertation

The work has been developed along seven chapters plus one of bibliography, which contains the reference links that have been used to document the different technologies used. These reference links will be shown through hyperlinks to the different sites. In the chapters the concepts marked in the summary of the following contents index will be deepened:

- **Chapter 1**. Introduction. It marks the beginning of the work and the motivation of it.
- **Chapter 2**. State of the art. It talks about the tools that have been used, the similar applications that we have been inspired by and the technologies used.
- **Chapter 3**. Methodology and technology. We will use the Kanban methodology to show the different phases of the project since its inception. We will also delve into the technologies used.

- **Chapter 4**. Application design. Final sample of the application, explanation of the interface, risk management and design of the same.
- **Chapter 5**. Architecture and implementation. The different layers of the application will be displayed.
- **Chapter 6**. Results. Discussion about the final results of the project.
- **Chapter 7**. Conclussions and future work. Conclussions and future work.
- **Chapter 8**. Bibliography.

Chapter one and seven can also be found in English as it is specified in the regulation for End-of-Degree Projects of the UCM.

Capítulo 2

Estado del arte

Se procede a hablar de herramientas de temática similar o que se podrían utilizar de manera similar, aplicaciones en las que nos hemos inspirado o con precedente histórico para realizar nuestra aplicación y tecnologías usadas para generar el producto final.

En este capítulo se tratará:

2.1. Estado del arte (explicación de herramientas calendario)

Se han trabajado con varios programas a lo largo del trabajo. Estos programas generan horarios con unas serie de restricciones. Unos están más orientados a generar aulas, a la gestión docente, etc, pero tienen una serie de restricciones específicas como los cambios de horario específicos del docente o algún cambio de horario, es decir, no tienen las restricciones blandas que son típicas del día a día.
Además, por lo general usan algoritmos de búsqueda local, o evolutivos; al final no se han utilizado estos tipos de programa porque se quiere que el programa se utilice en dispositivos móviles; y ambos métodos acaban usando mucha memoria.
Con la gestión de horarios lo que se pretende hacer es organizar las actividades durante las clases y fuera de ese horario. Por eso, nuestra herramienta agiliza ese proceso.

2.2. Herramientas calendario

Hay una gran variedad de gestores de horarios, tanto de empresa como para colegios.Durante el trabajo hemos encontrado las siguientes herramientas de generación de horarios:

- Aplicación coldideas.
- Canva.
- ascHorarios.
- Timetableweb.
- Doodle.
- Woffu.
- smartsheet.
- Gestión ucmnet.

2.2.1. Aplicación coldideas

Para poder utilizar esta aplicación nos meteremos en la página web [1], y veremos lo siguiente nada más entrar:

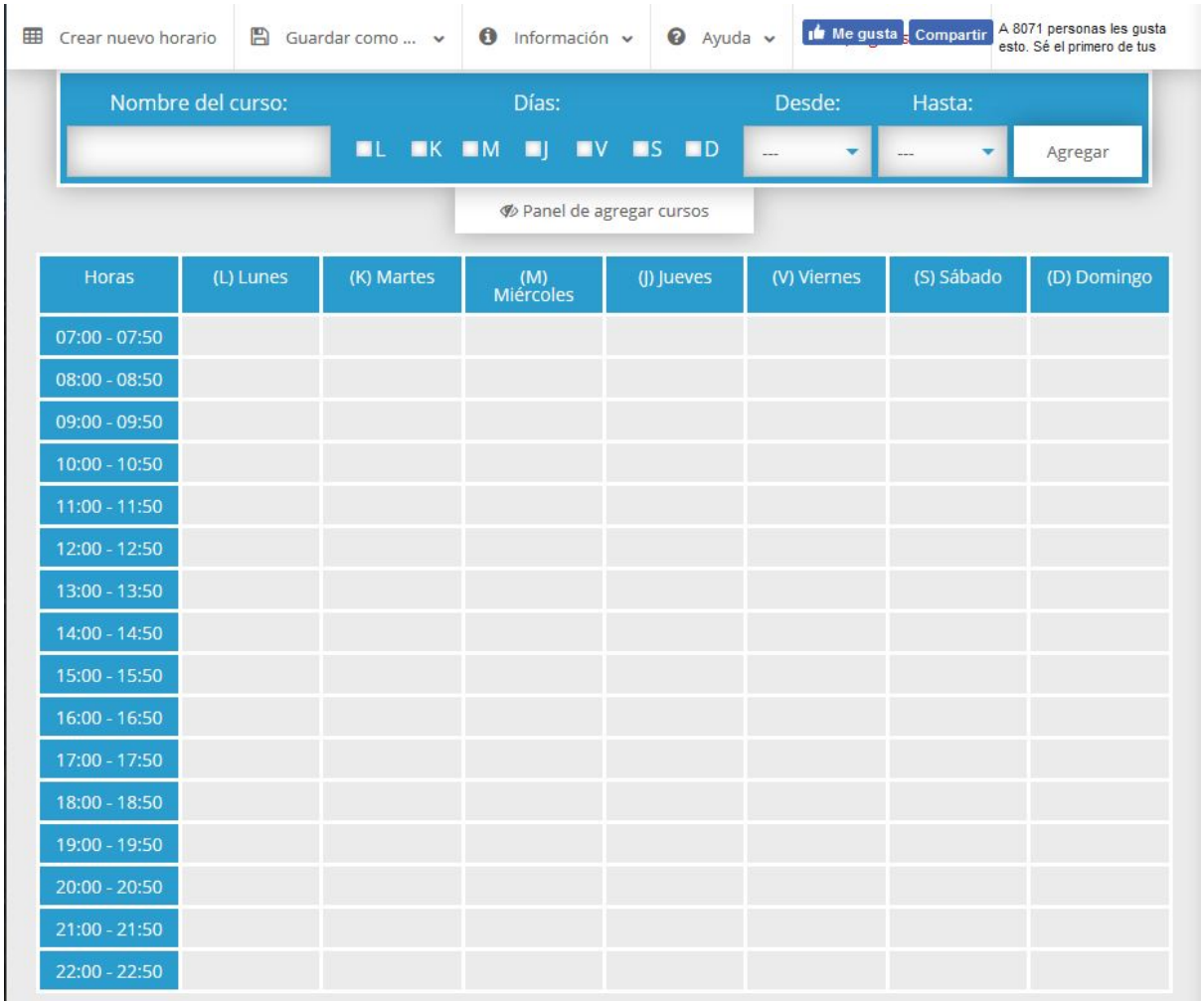


Figura 2.1: Captura de la aplicación de coldideas

Una vez nos hemos metido en la aplicación nos permitirá añadir el nombre de la asignatura con el nombre del curso, seleccionando los días y finalmente, nos permitirá agregar la asignatura.
Seguidamente, vamos a proceder a crear un horario de ejemplo y vamos a discutir sus limitaciones:

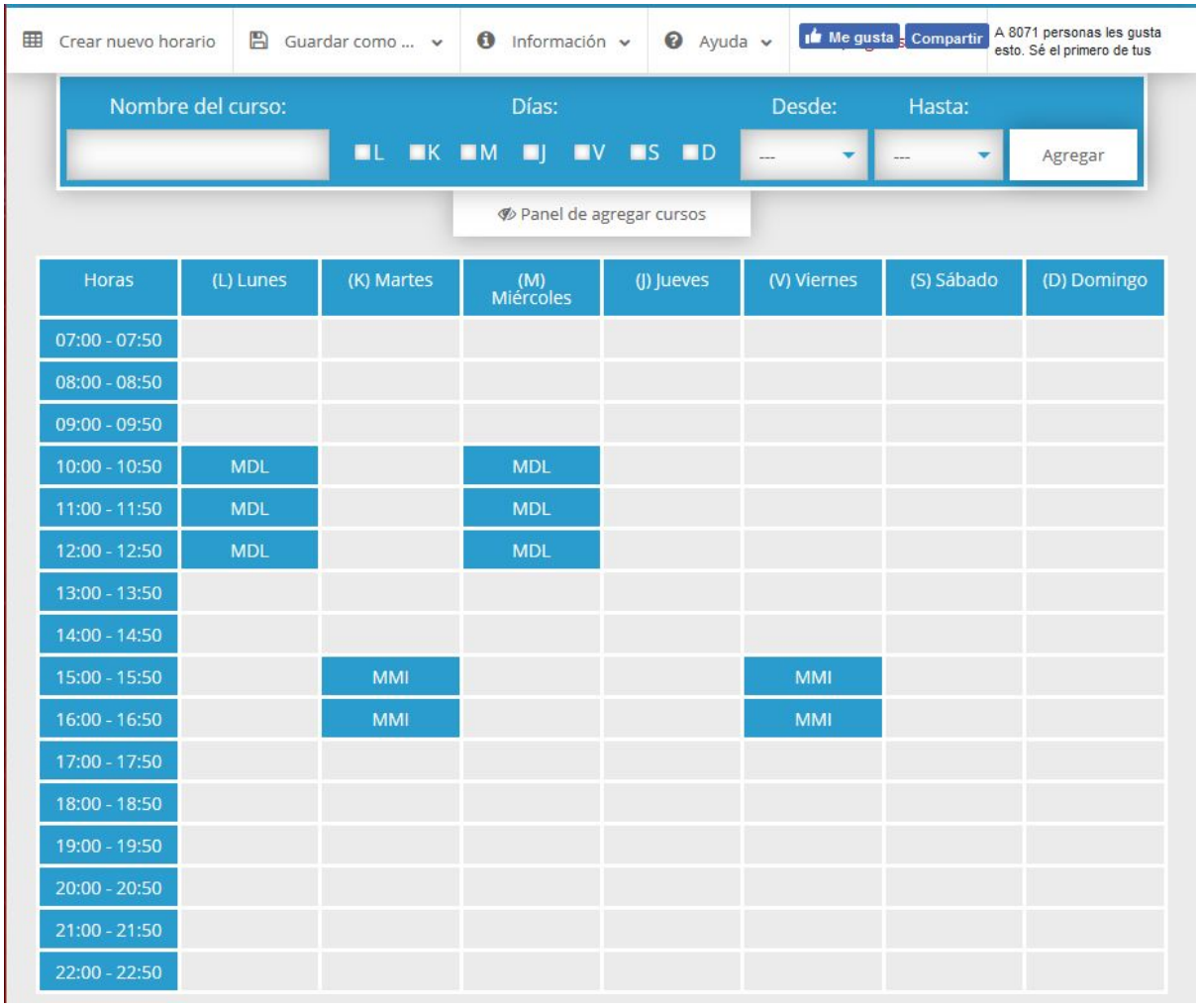


Figura 2.2: Captura de horario de la aplicación de coldideas[1]

Finalmente, entre las limitaciones que encontramos en la aplicación son las siguientes:

- Horario cerrado.
- Dificultad en la selección de los días.
- Dificultad para guardar los horarios.
- El usuario tiene que saber el horario previamente.

2.2.2. Canva

En esta aplicación nos permitirá seleccionar una serie de plantillas, en las cuales el usuario tendrá que cambiar a mano el nombre de la plantilla y las distintas anotaciones. Vamos a proceder a ver un ejemplo y ver sus desventajas:

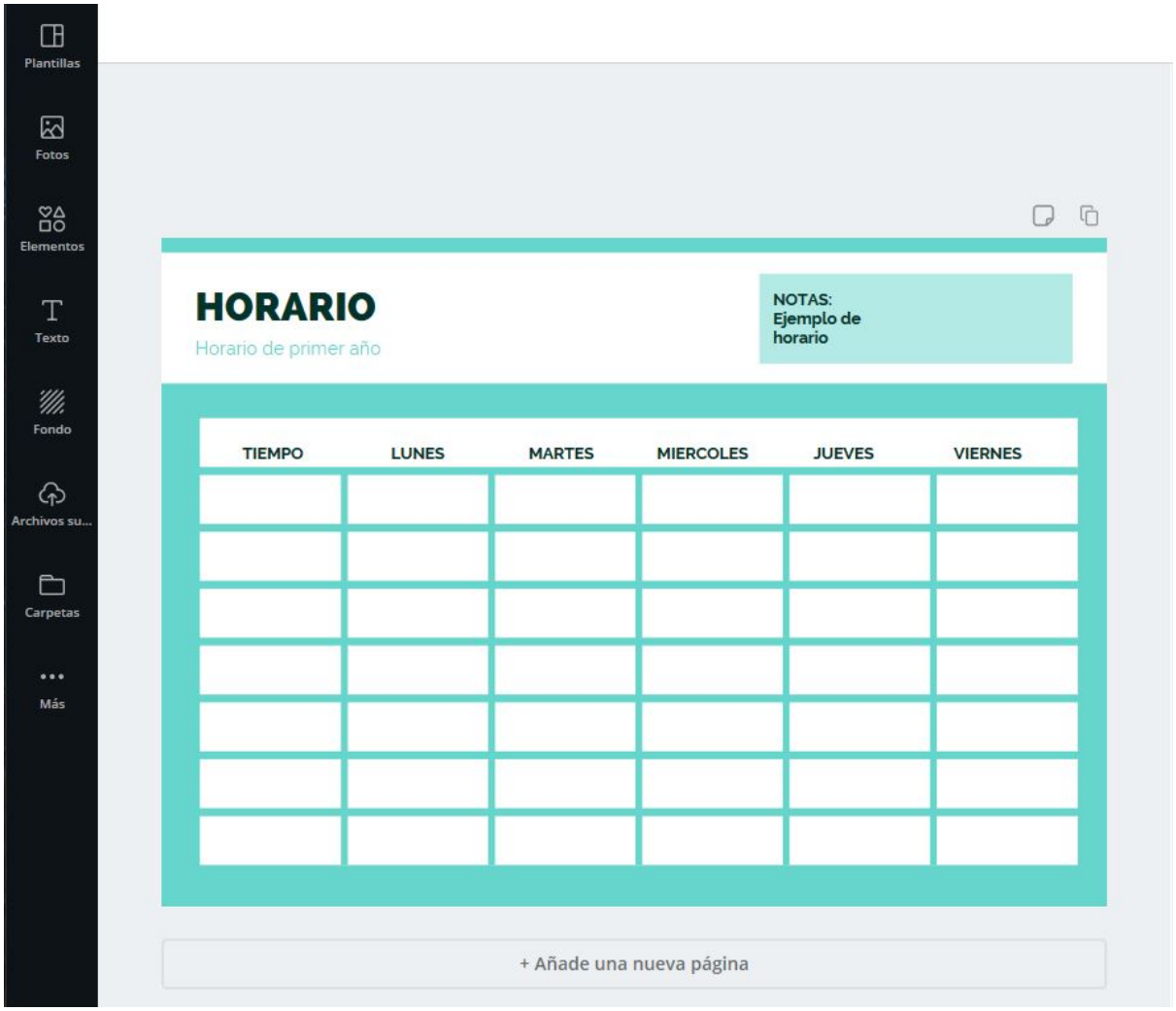


Figura 2.3: Captura de la plantilla de Canva[2]

Entre las desventajas que podemos encontrar utilizando esta aplicación son las siguientes:

- El usuario tiene que saber el horario previamente.
- Toda la realización del horario es manual.
- Es complicado acceder a la aplicación vía móvil.

2.2.3. ascHorarios

ascHorarios es un programa que se puede descargar a partir de la web [3], este programa tiene las funciones siguientes:

- Generación automática del horario.
- Permite realizar ajustes manuales.
- Verifica el horario mediante algoritmos.
- Introducción de datos de manera sencilla.

- Publicación de datos a través de dispositivos móviles.
- Permite importar los datos.
- Permite personalizar los horarios.
- Es específico para cada país.

Es importante indicar que este programa necesita instalación previa. Una vez hayamos instalado el programa, nos permitirá crear el horario de la siguiente forma:

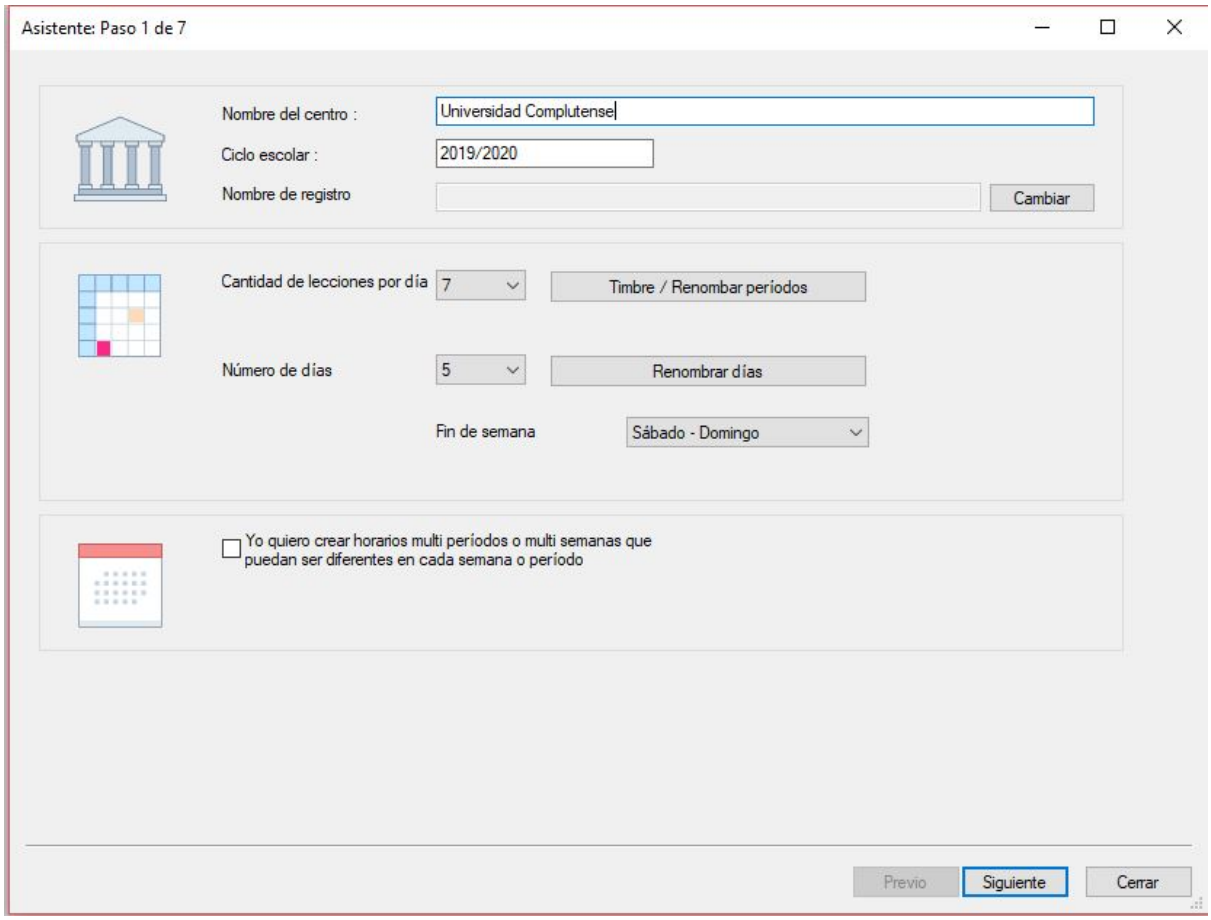


Figura 2.4: Captura de creación de horario con ascHorarios

Como podemos observar, primero tenemos que indicar el nombre del aula con las horas, seguidamente seleccionaremos las asignaturas y profesores que van a dar esas asignaturas. Una vez que hemos realizado los pasos anteriores veremos el siguiente horario:

Figura 2.5: Captura de horario creado en ascHorarios[3]

Frente a nuestra aplicación este programa tiene las siguientes desventajas:

- Sólo lo pueden utilizar los profesores.
- Tiene opción de pago para mejoras.
- Tiene una interfaz poco intuitiva.
- Es complicada de utilizar.

2.2.4. Timetableweb

Es una aplicación que sólo puede utilizarla los profesores, nos tendremos que registrar previamente antes de poder utilizarla y nos permitirá realizar el siguiente horario:

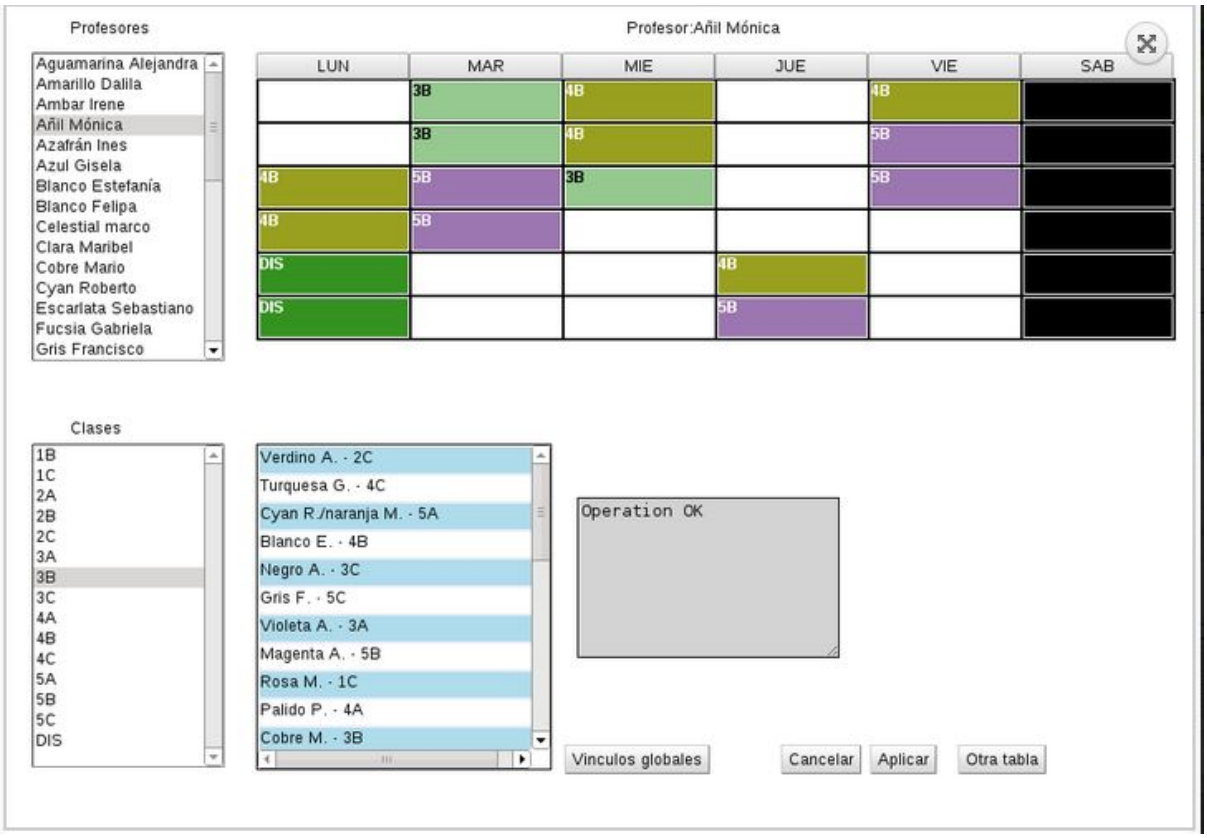


Figura 2.6: Captura de horario de timetableweb[4]

Frente a nuestra aplicación encontramos las siguientes desventajas:

- Sólo puede ser utilizado por el profesorado.
- Requiere registro previo.
- Es complicado de utilizar.
- No permite flexilizar los horarios.

2.2.5. Doodle

Doodle es una aplicación más conocida por la realización de encuestas que de horarios, pero nos permite realizar pequeños horarios y compartirlos con los usuarios. Cuando empezamos a realizar un horario veremos lo siguiente:

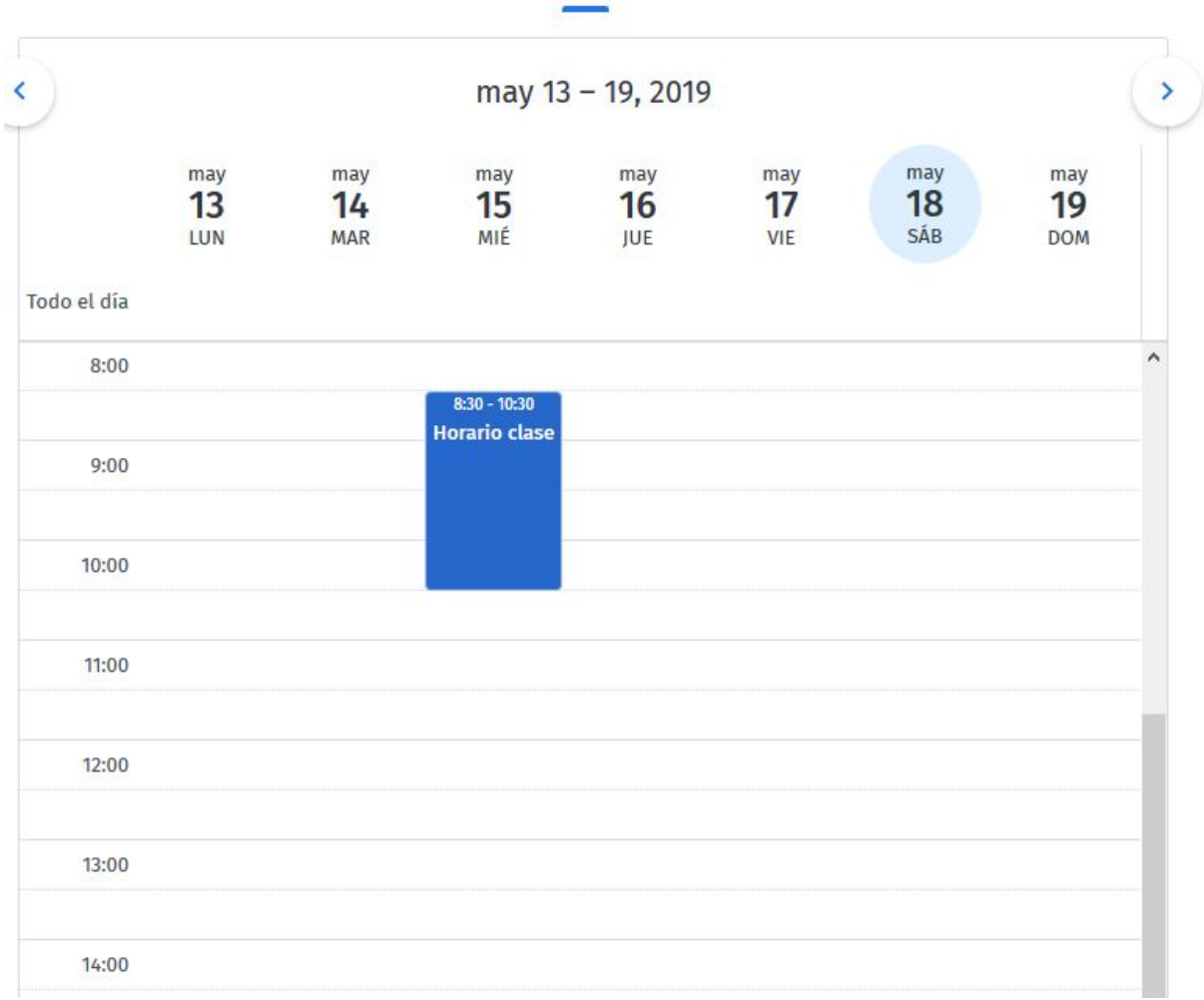


Figura 2.7: Captura de horario de Doodle[5]

Y frente a nuestra aplicación encontraremos las siguientes desventajas:

- Necesita registro previo.
- Sólo nos permite organizar una actividad.
- Es complicada de utilizar.
- Te obliga a compartir el horario.

2.2.6. Woffu

Woffu es una aplicación utilizada en el mundo laboral, en ella manejaremos el horario de nuestros trabajadores. Vamos a ver un ejemplo de horario:

Figura 2.8: Captura de horario de Woffu[6]

Como hemos visto con las anteriores aplicaciones vamos a ver las desventajas respecto de nuestra aplicación:

- Sólo nos permiten seleccionar los días ocupados.
- Está permitido sólo su uso para empresas.
- No lo pueden utilizar ni alumnos ni profesores.

2.2.7. smartsheet

smartsheet es una dirección web que nos permite planificar nuestro horario semanal o diario. Lo malo es que sólo accederemos a una versión gratuita. Vamos a ver un ejemplo de horario:

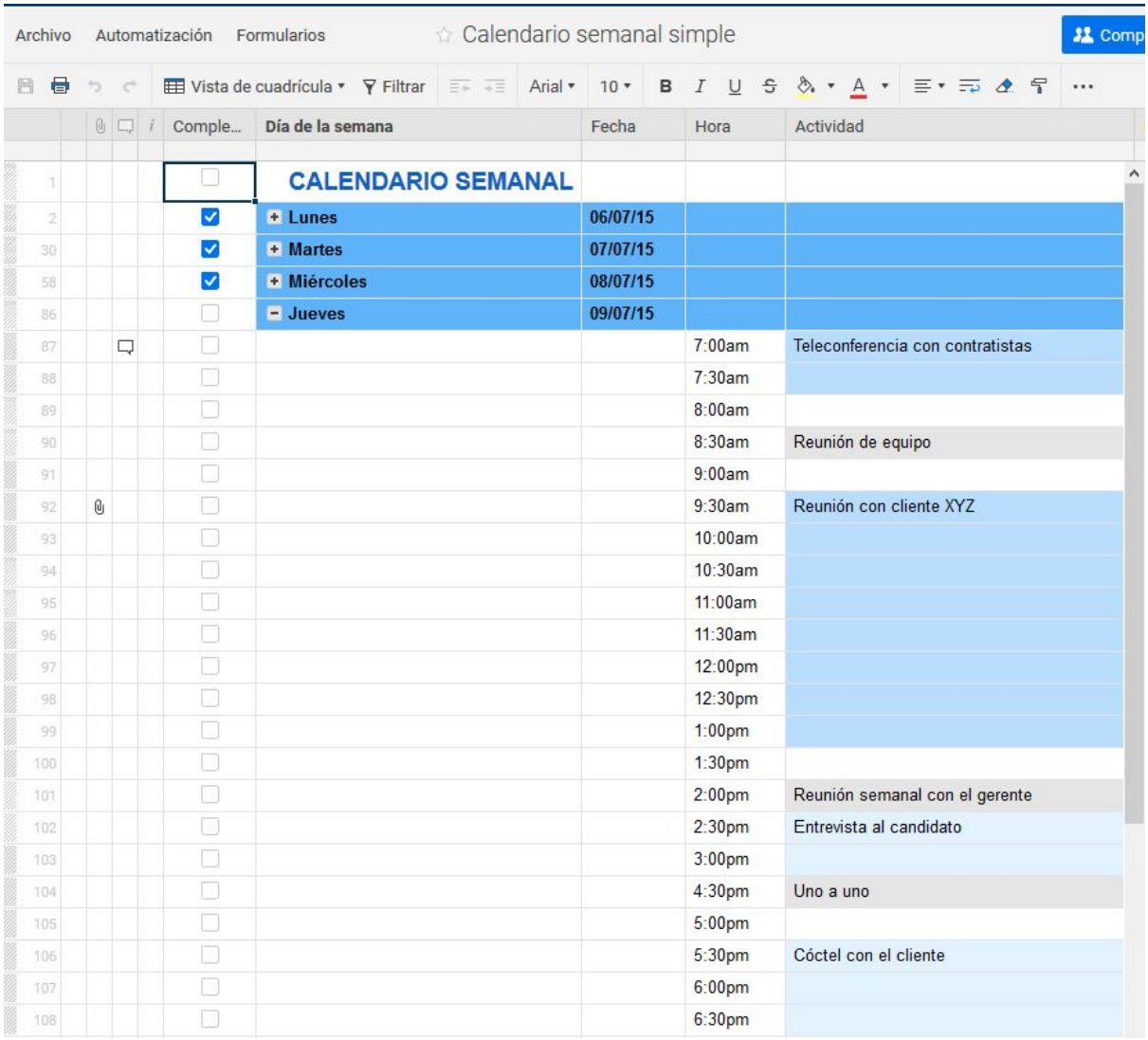


Figura 2.9: Captura de horario de smartsheet[7]

Vamos a ver las desventajas respecto a nuestra aplicación:

- Es una versión de prueba.
- Hay muchos tipos de horarios.
- No es para uso académico.

- Su uso es complejo.

2.2.8. Gestión ucmnet

Por último, tenemos el programa de gestión de ucmnet. Este sistema de gestión nos permite, una vez hemos elegidas las asignaturas, crear nuestro horario o mirar cómo quedaría este en el momento que hemos realizado nuestra matrícula. Cuando entramos en nuestro sistema de gestión veremos lo siguiente:

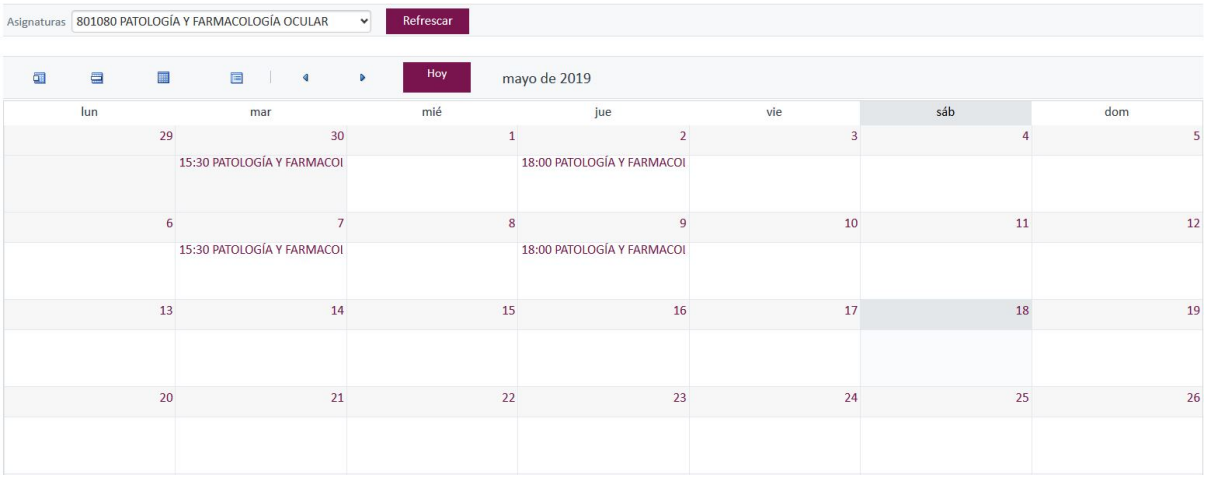


Figura 2.10: Captura de horario de Geaportal[8]

Tiene las siguientes desventajas respecto a nuestra aplicación:

- La selección de asignaturas y grupos en manual.
- Una vez hemos terminado la selección, la aplicación mostrará si se solapan o no las asignaturas. En caso de estarlo, hay que volver atrás y probar otra combinación.
- La gestión del horario es complicada.
- El manejo de los distintos horarios es complicado.

Capítulo 3

Metodología y tecnología

Basándonos en un proceso ágil de desarrollo vamos a mostrar las fases del proyecto desde la reunión inicial. También vamos a mencionar con profundidad las tecnologías usadas para esta parte.

En este capítulo se tratará:

3.1. Metodología usada y explicación

En esta sección explicaremos qué metodología preferimos utilizar para llevar a cabo el proyecto y en qué consiste.

3.1.1. Metodología Kanban

Hemos elegido este método porque se realiza el trabajo de una manera fluida, ya que, nos permite distribuir el flujo de trabajo.
Esta metodología está representada a través de la tarjeta Kanban, con ella extraemos la información de nuestro trabajo y la vamos poniendo a través de un entorno de trabajo.
Vamos a proceder a mostrar un ejemplo sencillo de esta metodología:

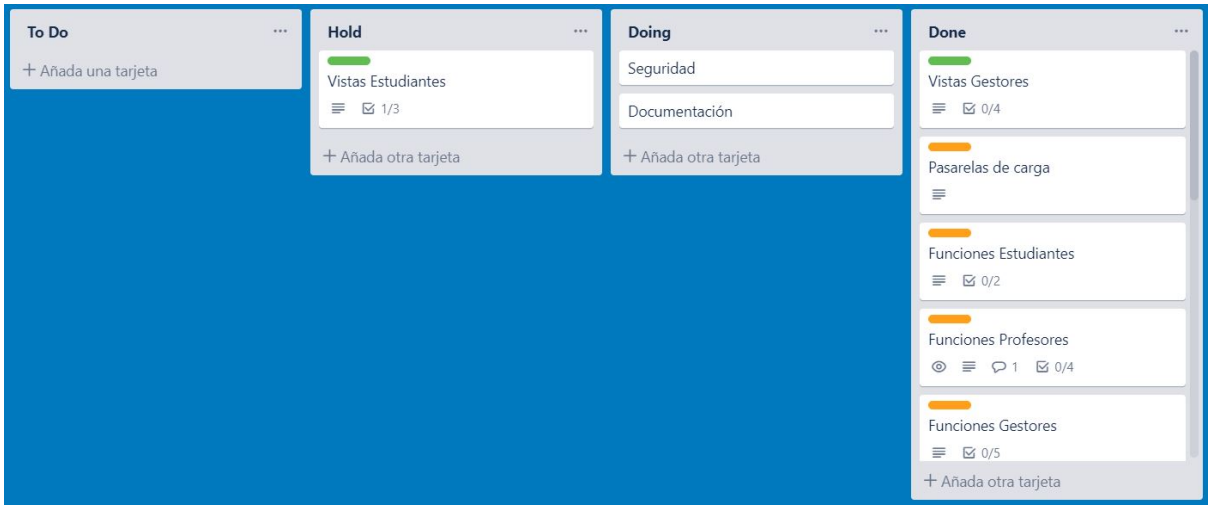


Figura 3.1: Metodología Kanban sencilla[9]

Seguidamente vamos a mostrar el funcionamiento de esta metodología:

- Visualización total del trabajo.
- Se mantiene un control del trabajo entre todos los miembros del equipo.
- Se realiza un seguimiento del tiempo de forma continúa y se puede evaluar el trabajo realizado.
- Con un vistazo podemos ver cómo llevamos nuestro trabajo.
- Se puede identificar con mucha facilidad cuando estamos atascados.

Con este método hemos visto aumentado nuestro rendimiento, ya que, nuestras tareas están distribuidas, controladas en todo momento y podemos ir realizando ajustes a lo largo del trabajo.
Posteriormente, hemos seguido estos 4 principios básicos de la metodología Kanban:

a). **Empezar con lo primero**: Para poder identificar los problemas.

b). **Mejora de los cambios**: Es mejor no realizar cambios radicales.

c). **La importancia de los roles**: Diferencia las responsabilidades de cada rol.

d). **La figura del líder**: Para mejorar el rendimiento se necesita un liderazgo claro.

Hemos encontrado los siguientes beneficios utilizando este método:

- **Medición precisa del rendimiento**: Nos permite estimar qué nivel de trabajo llevamos.
- **Mejor organización**: Nos permite ver ver mediante columnas, carriles y colores cómo está trabajando el equipo y la colaboración en tiempo real.
- **Distribución del trabajo**: Nos permite visualizar de una manera constante el flujo de las tareas.

3.2. Plan de proyecto

Antes de comenzar el proyecto se tuvieron varias reuniones con el tutor en las cuales se acordó una división en dos etapas del proyecto. Estas dos etapas serían la de documentación de las tecnologías debido al desconocimiento de muchas de ellas por parte del equipo desarrollador y la fase de implementación, que evidentemente incluye los diseños de las vistas, las entrevistas a usuarios y gestores, y las funciones de los mismos dentro de la aplicación.

- Documentación de las tecnologías.
- Planteamiento inicial del proyecto.
- Diseño de la aplicación por Kanban.
- Desarrollo y pruebas.
- Documentación de la aplicación.

3.2.1. Documentación de las tecnologías

Durante esta fase se hizo un estudio detallado de la documentación y se decidió utilizar el lenguaje PHP[10] como lenguaje de programación para el backend, para el frontend se utilizó HTML, Javascript, jQuery[11] para las librerías, jQuery-ui[12] para añadir funcionalidad y dinamismo al contenido. Además se utilizó el código CSS[13] y Javascript[14] de Bootstrap para reducir la cantidad de tiempo a invertir en diseño de interfaces y funcionalidades. Además, queríamos que toda nuestra aplicación se desarrollara con licencia libre. Este proceso fue bastante sencillo ya que, había documentación de todas las tecnologías citadas anteriormente.

3.2.2. Planteamiento inicial del proyecto

Una vez visto con qué tecnologías podríamos desarrollar la aplicación, comenzamos a ver cómo podemos crear nuestra aplicación. Para reducir el tiempo en el desarrollo de la interfaz hemos utilizado CSS y Javascript de Bootstrap. Al principio, las premisas que planteamos es que fuera una aplicación con una interfaz sencilla de utilizar y que fuera rápida de procesar. La estructura que se planteó es que fuera sencilla de utilizar, que primero pudiéramos seleccionar la carrera que estamos haciendo con su respectivo

itinerario, y que fuéramos añadiendo las asignaturas, y se fueran colocando cuando le diéramos a un botón. Pudiendo así ver en el momento todo nuestro horario. Se decidió no hacer que el horario se calculara automáticamente cada vez que se añadiera una asignatura. Reduciendo así el número de conexiones al servidor y la carga en el dispositivo del usuario.

3.2.3. Diseño de la aplicación por Kanban

Lo primero que hicimos antes de empezar a planificar nada fue abrirnos una cuenta en Trello [15], una aplicación web que permite que varias personas compartan un tablero Kanban almacenado en la nube. Además, tiene una aplicación para dispositivos móviles que te permite ver cómo de avanzado van tus compañeros.
Al crear un tablero en Trello, este está vacío. Tan solo te ofrece la opción de comenzar a insertar columnas. Esto nos pareció una idea genial, ya que podríamos poner tantas etapas a nuestro proyecto como viéramos necesarias. Tras pensarlo, decidimos que solo podríamos cinco: «To-Do», para aquellas tareas que estaban pendientes de ser realizadas; «Doing», para aquellas que se estaban realizando en ese momento; «Hold», para aquellas que no se podían continuar hasta obtener los resultados de otra tarea; «Dropped», para las tareas que se acaban descartando; y «Done», para las tareas que han sido realizadas.
Una vez creada la estructura, dimos comienzo a una fase de atomización. En un papel, empezamos a escribir las tareas que hay que realizar como, por ejemplo, hacer el formulario de la página de inicio. Una vez elaborada la lista de tareas, se intentaba dividir estas en otras más pequeñas. Así una y otra vez hasta que llegamos a un listado de tareas atómicas, fáciles de realizar, cuyo tiempo de realización era pequeño y que nos facilitaba saber en qué punto del proyecto nos encontrábamos. Fue en este punto donde decidimos que solo habría tres roles de usuario.

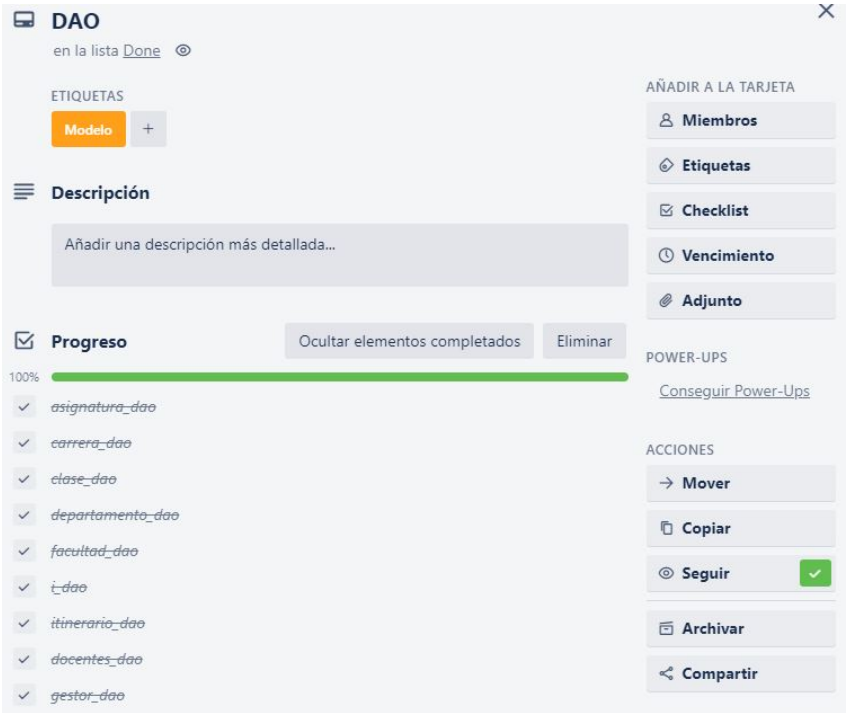


Figura 3.2: La tarjeta correspondiente a los DAO

Con la lista volcada en Trello, comenzamos a asignarles un sencillo código de colores a las tarjetas: verde para las tarjetas relacionadas con el frontend, naranja para las relacionadas con el backend y sin color para las tarjetas relacionadas con ambas partes.
Lo primero que hicimos una vez estaban todas las tareas preparadas, fue empezar por el pila fundamental de la aplicación, los horarios. Ha sido la tarjeta que menos tiempo ha tardado en pasar de «Doing» a «Hold». Mientras esperábamos a que nos llegara el horario en un formato que fuera procesable, nos pusimos a diseñar las vistas.

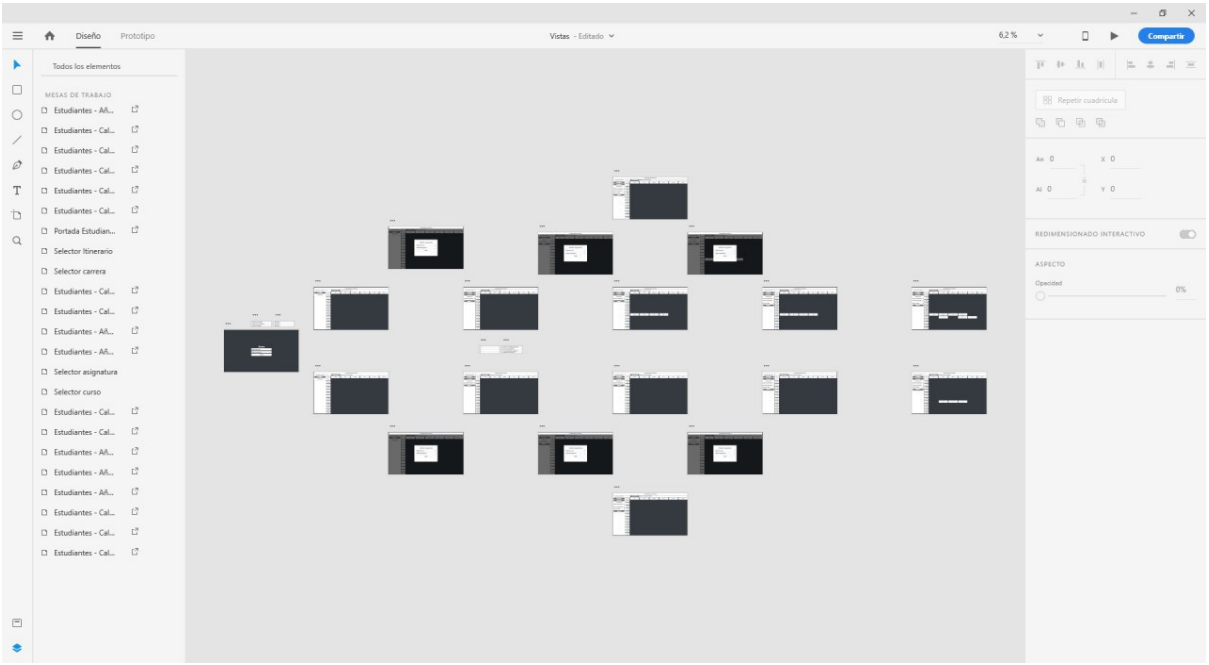


Figura 3.3: Captura de pantalla de Adobe Xd

Ya con las vistas diseñadas, y con un fichero de horarios con el que poder trabajar, comenzamos a mover las tarjetas.

3.3. Tecnologías usadas

En esta sección se verán las tecnologías usadas para desarrollar el proyecto.

3.3.1. PHP

PHP[10] proviene del acrónimo de Hypertext Preprocessor, es un lenguaje en código abierto que se utiliza para desarrollo web.
Las páginas en PHP contienen código en HTML incrustado.
Hemos decidido utilizar este código para programar nuestro backend porque es un código simple de utilizar y porque está centrado en los scripts. Además nos ha permitido utilizar una base de datos para almacenar todas las asignaturas seleccionadas.

3.3.2. CSS

CSS[13] es un lenguaje que da estilo al documento HTML. Describe cómo los elementos HTML se deben disponer.

3.3.3. jQuery

jQuery[11] simplifica el uso de Javascript al encapsularlo en funciones capaces de interactuar con prácticamente cualquier parte del frontend. También soporta conexiones asíncronas, las cuales se utilizan en muchos casos en esta aplicación, ya que evitan que se tenga que recargar la página con cada acción que se quiera realizar.

3.3.4. jQuery-ui

jQuery-ui[12] nos permite habilitar interacciones en la interfaz, accesorios y temas para jQuery. Además nos permite ejecutar los buscadores en cualquier servidor web.

3.3.5. Bootstrap

Bootstrap[14] nos permite reducir el tiempo en diseño mediante el uso de una intuitiva batería de clases CSS. Tan solo hay que declarar que un elemento es de una clase CSS que esté en dicha batería para que se aplique el estilo asociado.

3.3.6. Overleaf

Es la plataforma[16] utilizada para la realización de la memoria de este TFG. Se ha realizado mediante la programación LaTeX.

Capítulo 4

Diseño de la aplicación

Muestra final de la aplicación, explicación de la interfaz, casos de uso, gestión de riesgos y diseño de la misma.

En este capítulo se tratará:

4.1. Descripción

La falta de organización de los horarios es un problema al que se enfrentan tanto los estudiantes como los docentes. Con Chronos hemos resuelto esa falta de organización, habilitando una aplicación que les permite planificar el año de una forma cómoda y sencilla.
Primero, permite a los estudiantes planificar los horarios realizando todas las combinaciones posibles de clases hasta encontrar una que se adapte a su disponibilidad. Se podrán beneficiar de esta aplicación, ya que, les permite planificar la matriculación de distintas asignaturas de cursos distintos con sus asignaturas optativas.
Con esta aplicación, el estudiante podrá ver al instante si las asignaturas en las que se quiere matricular junto con su disponibilidad son compatibles. Así, podrá conocer si es posible que pueda llevar a cabo el curso académico previsto.
Seguidamente, los docentes podrán organizarse a la hora de escoger las asignaturas que impartirán.
Una vez registradas las asignaturas y el número de docentes que puede tener cada una, se pasa a registrar las preferencias de cada uno de los docentes. Una vez están registradas estas, se le proporciona un fichero al asistente en el que se indica el orden de elección. Es el propio asistente, con el orden indicado en el fichero y las preferencias de cada uno, el que asigna las plazas. En caso de que una asignatura no tenga más plazas disponibles, se intentará asignar al docente en la siguiente asignatura que había en su lista de preferencias.

4.2. Puntos a tratar

En las siguiente secciones detallaremos el diseño general de la aplicación, los ejemplos de uso y la gestión de riesgos activa.

4.3. Diseño general

Lo primero que vemos nada más entrar en la aplicación es la selección de carrera e itinerario. Veríamos lo siguiente:

Figura 4.1: Interfaz de inicio

Finalmente, una vez que hemos seleccionado la carrera con su itinerario veremos lo siguiente:

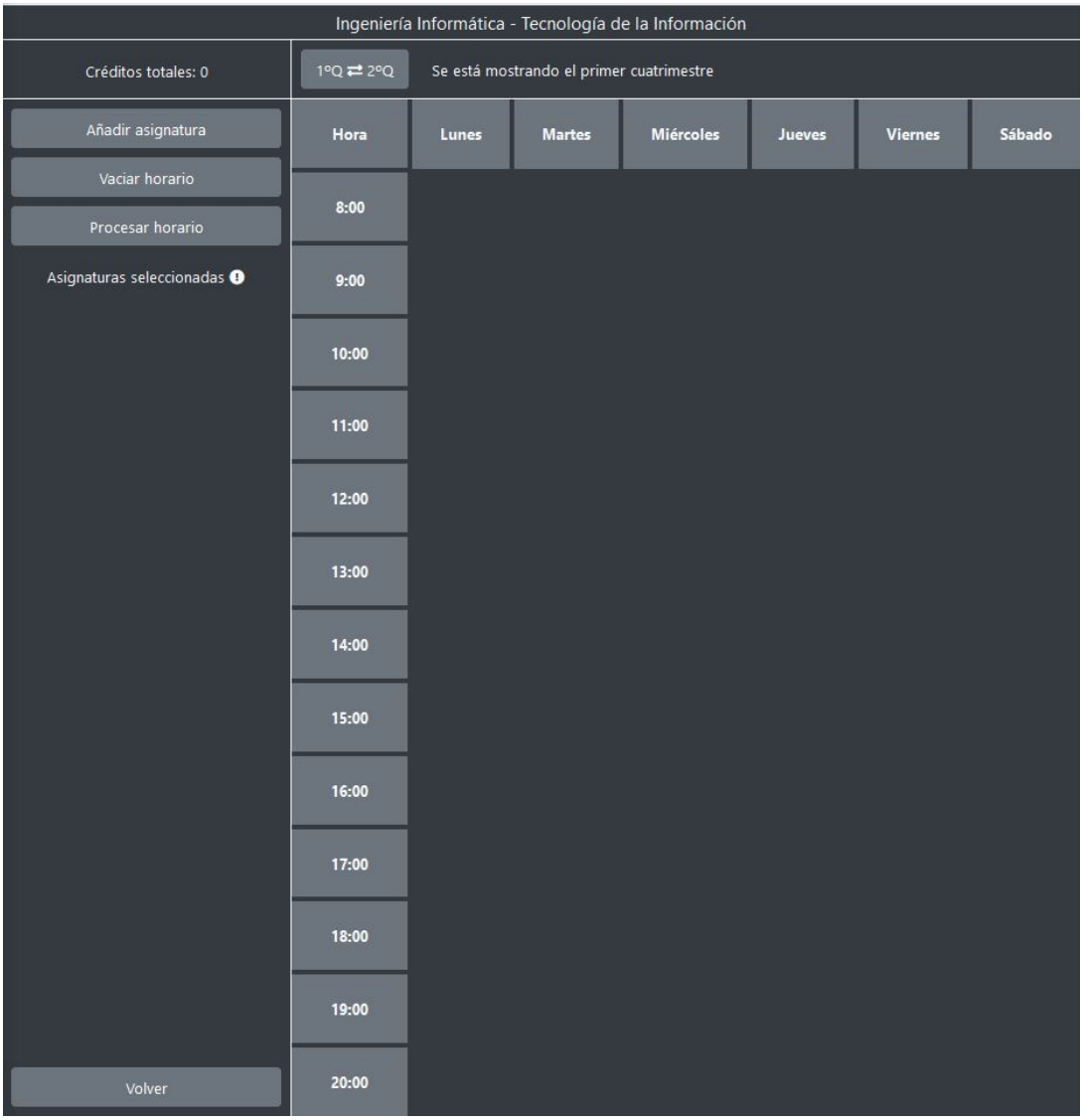


Figura 4.2: Horario Chronos

En esta interfaz se nos permitirá realizar nuestro horario, podremos seleccionar las asignaturas, vaciar el horario y una vez lo hemos terminado, procesaremos el horario.

4.4. Ejemplo de uso

En este apartado vamos a ver cómo utilizar la aplicación, dependiendo de los tipos de usuarios:

- Estudiantes.
- Docentes.
- Gestores.

4.4.1. Tipos de usuario

A continuación, procederemos a exponer cada uno de los roles de usuario que hay en esta aplicación.

4.4.2. Estudiantes

Son los que están interesados en conocer cómo se va a organizar el próximo curso, ya que, si tienen asignaturas de otros años u optativas puede haber incompatibilidades de horario.
Lo primero que verán nada más entrar en la aplicación son dos pestañas desplegables, una en la que seleccionen la carrera que están cursando junto con su itinerario, y podrán empezar a realizar su horario.

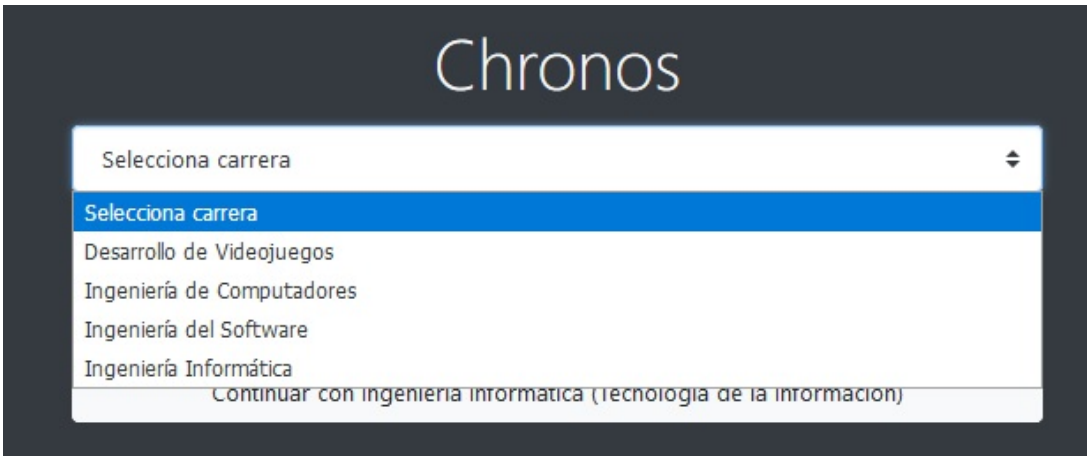


Figura 4.3: Selección de carrrera

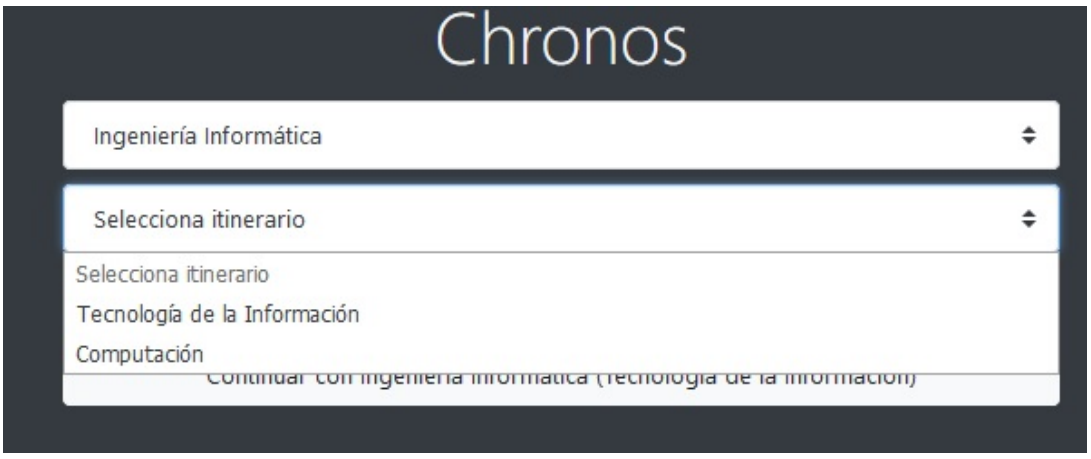


Figura 4.4: Selección de itinerario

Además de permitir elegir al estudiante elegir tanto carrera como itinerario, también le permite seguir con el horario que ha realizado previamente.

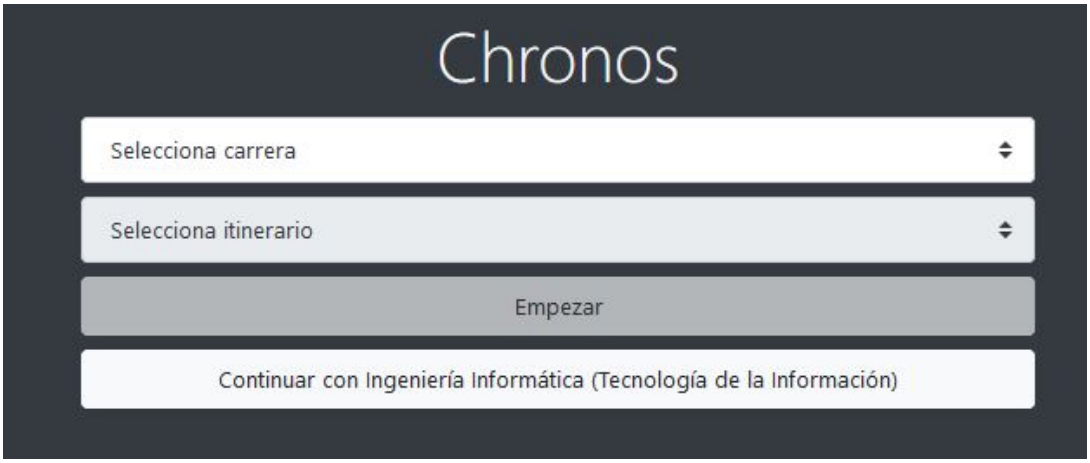


Figura 4.5: Volver a realizar un horario previo

Una vez que el alumno ha seleccionado tanto la carrera como la iteración, el alumno irá añadiendo distintas asignaturas tanto en el primer cuatrimestre como en el segundo, y procesar el horario.
A continuación vamos a ver cómo el alumno puede seleccionar las asignaturas que va a cursar:

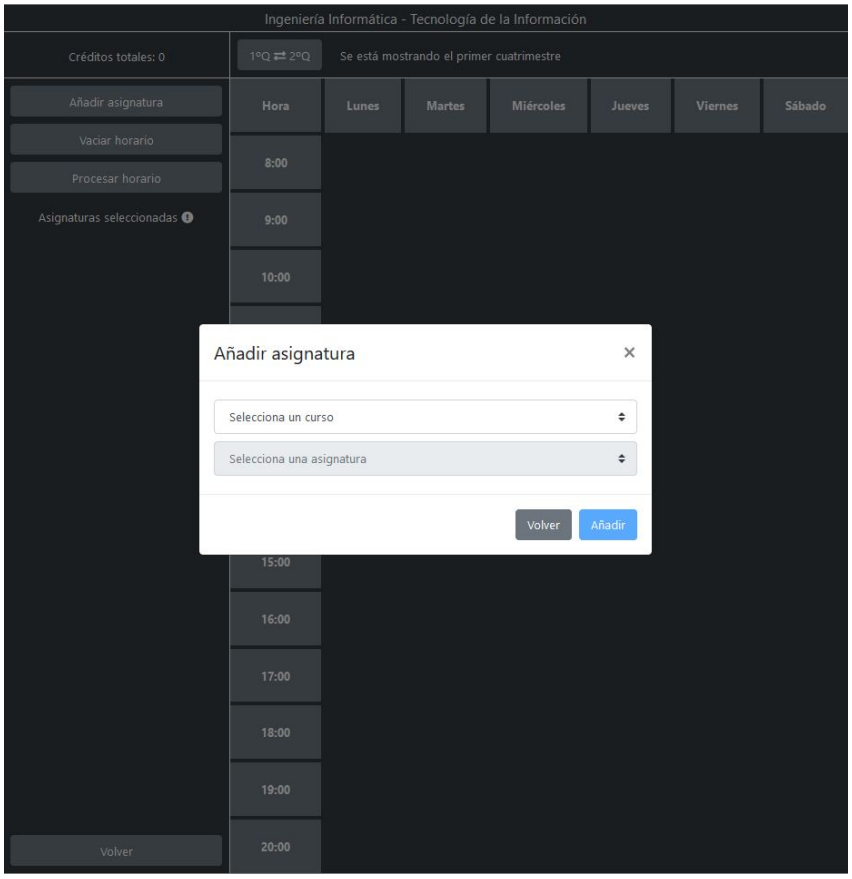


Figura 4.6: Añadir asignaturas

El alumno irá añadiendo las asignaturas y aparecerán a la derecha, con los créditos de cada una de las asignaturas y le aparecerán los créditos totales en su horario. Un ejemplo de horario de un alumno sería el siguiente:

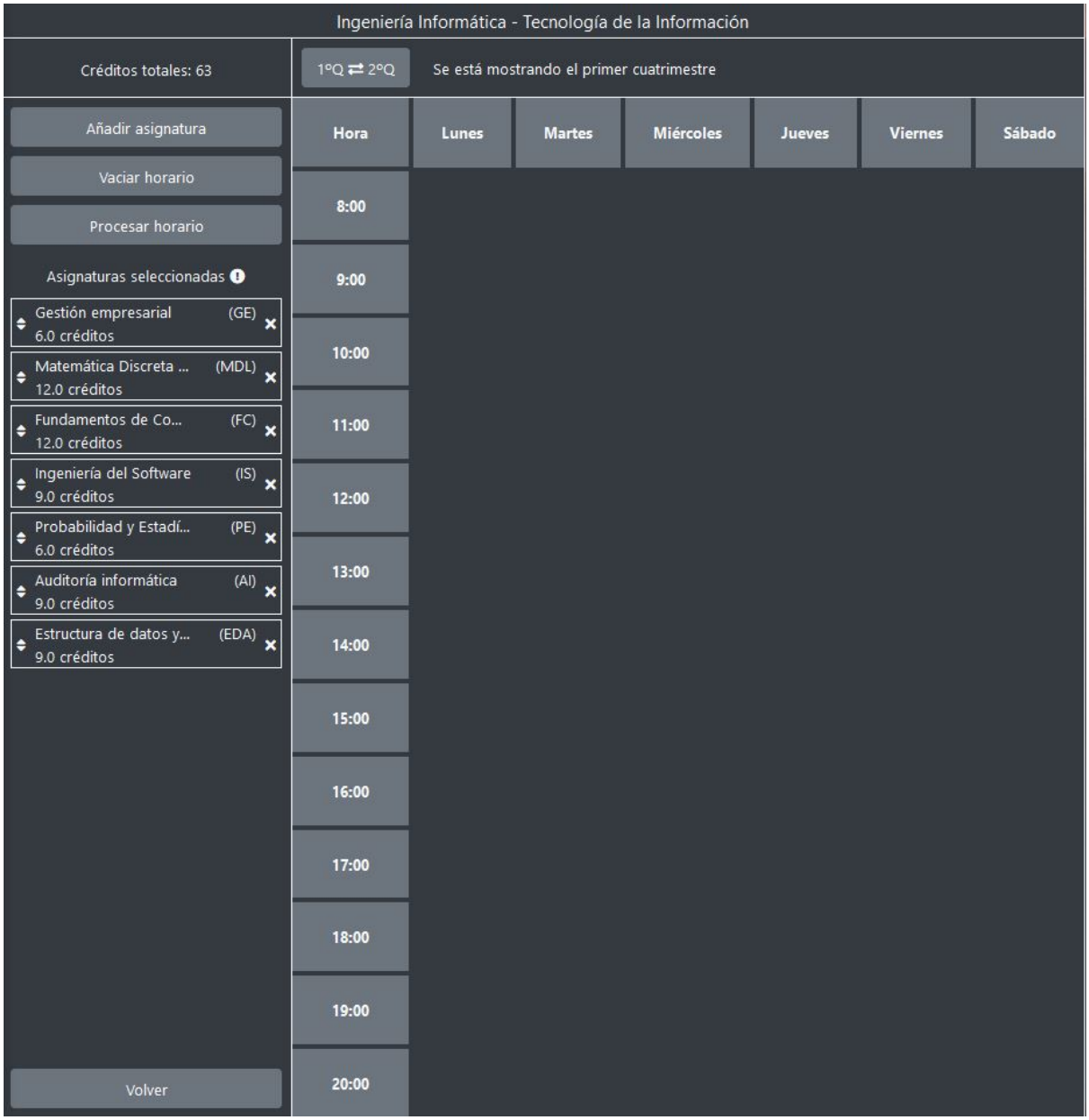


Figura 4.7: Ejemplo de horario de alumno

Una vez el alumno ha seleccionado las asignaturas que quiere cursar tendrá que ver si son compatibles con su horario, para ello utilizará la pestaña de Procesar horario. El alumno verá lo siguiente:

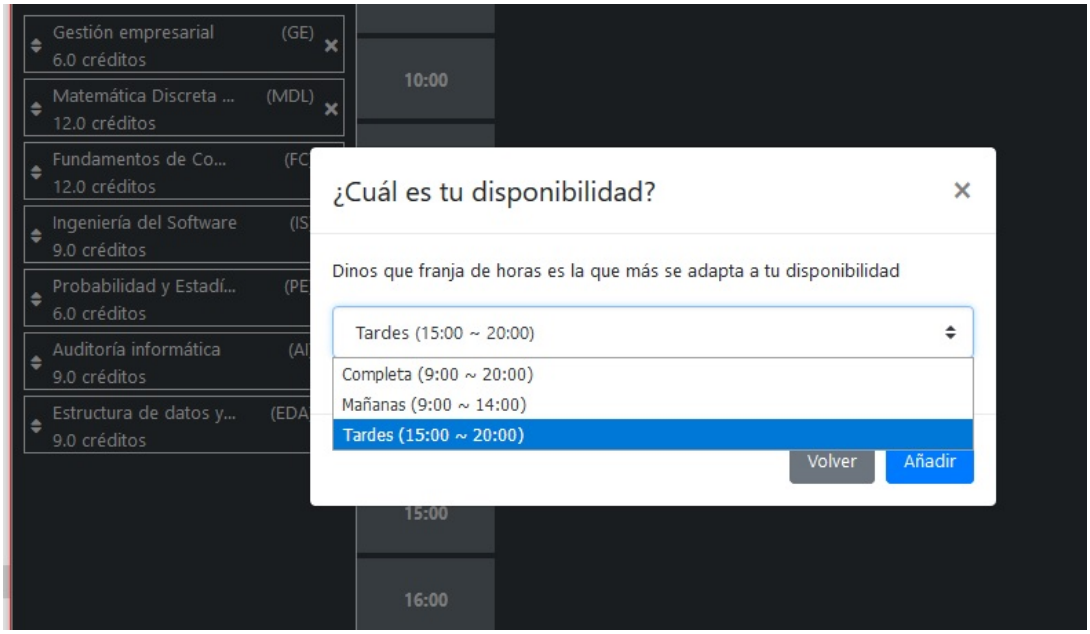


Figura 4.8: Disponibilidad

Una vez que el alumno ha seleccionado su disponibilidad, sólo le tendrá que dar a añadir y Chronos le indicará si puede realizar ese horario o no.
Finalmente, el alumno podrá cambiar entre el primer y el segundo cuatrimestre para ver si su horario se adapta a la disponibilidad.

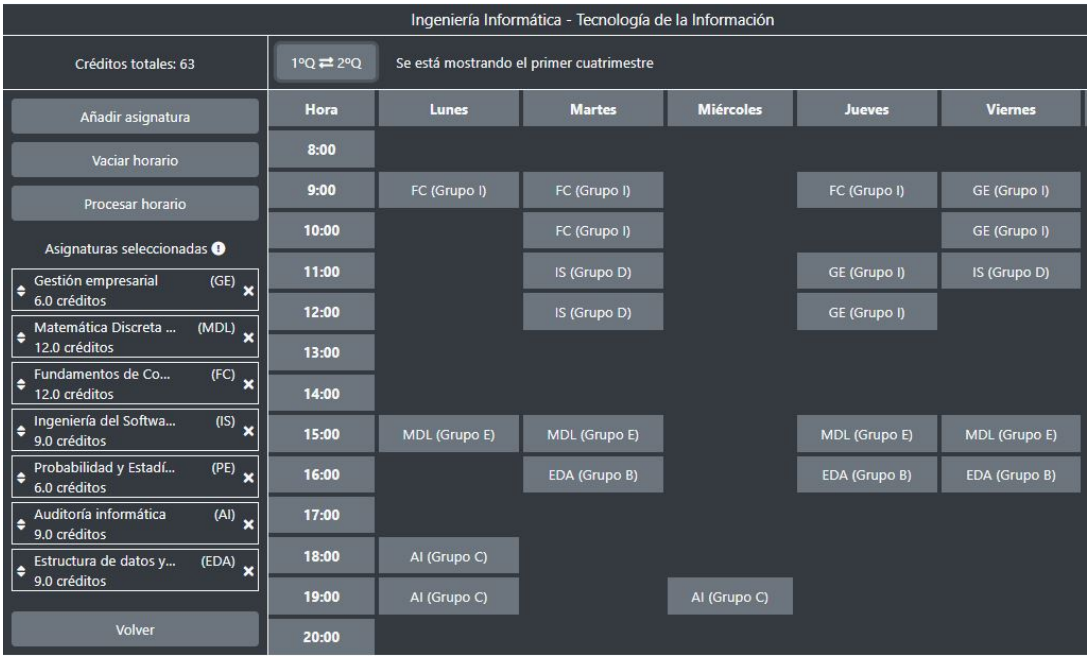


Figura 4.9: Horario del primer cuatrimestre

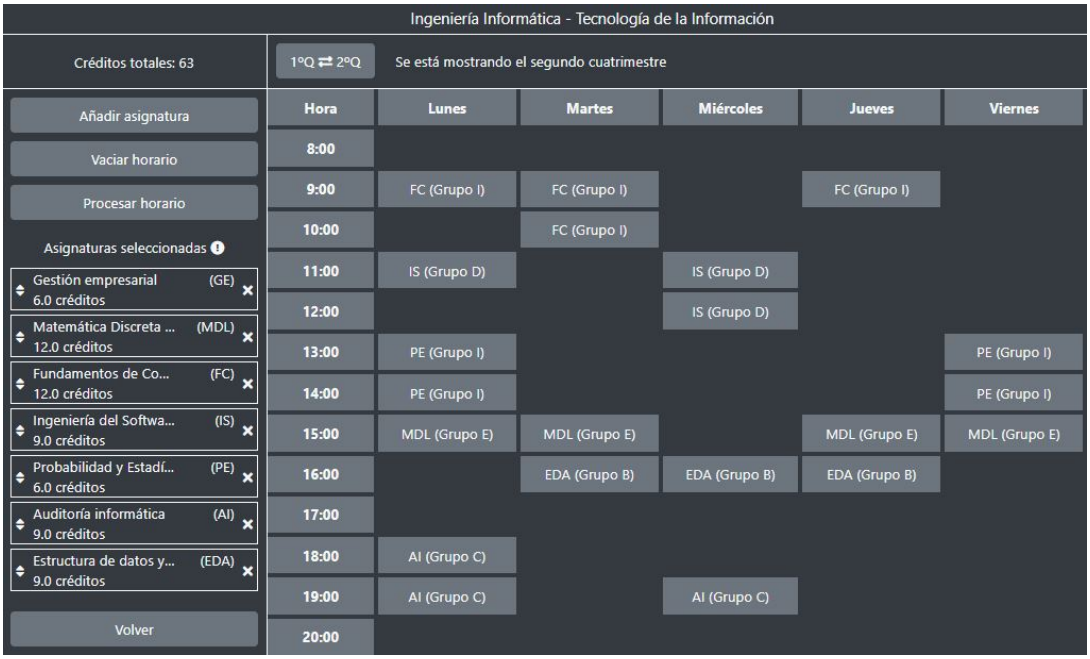


Figura 4.10: Horario del segundo cuatrimestre

4.4.3. Docentes

Seguidamente, vamos a ver cómo pueden utilizar nuestra aplicación los docentes.
Para los docentes, al contrario que los alumnos necesitarán un usuario y una contraseña para poder acceder a la aplicación. Esta será la página de inicio de los docentes:

Figura 4.11: Páginia inicio docentes

Una vez que inicien sesión, verán la lista de preferencias de asignaturas a elegir y podrá guardarlas. El docente verá lo siguiente nada más entrar en la aplicación:

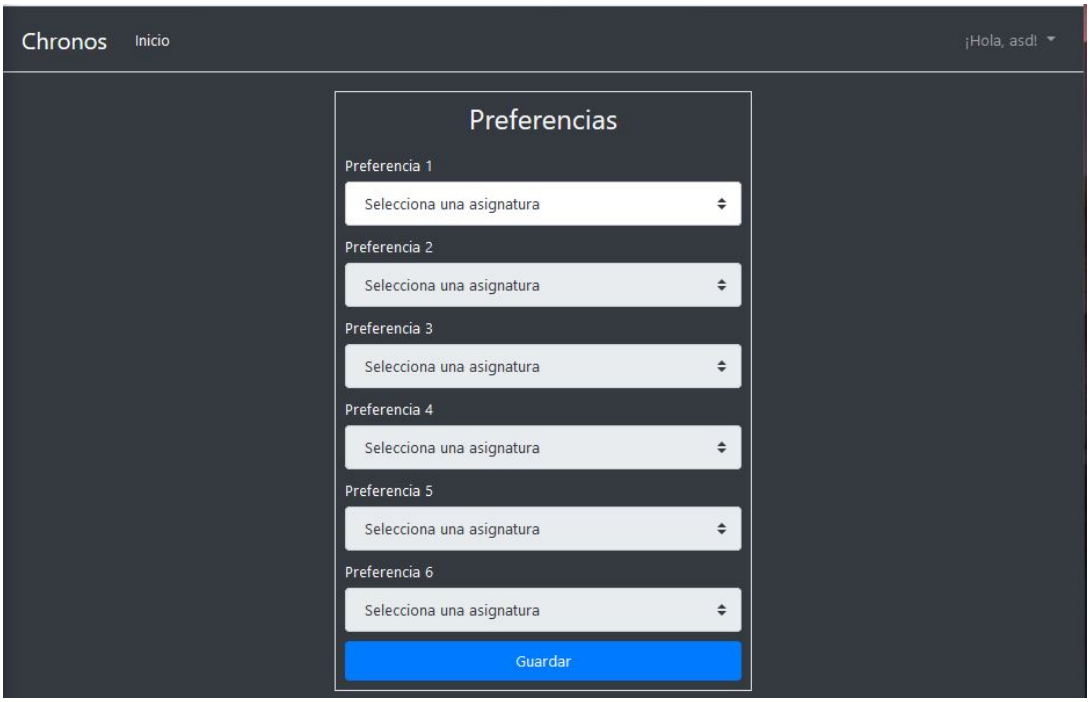


Figura 4.12: Página inicio docentes

Y una vez seleccionen una asignatura de preferencia verán lo siguiente:

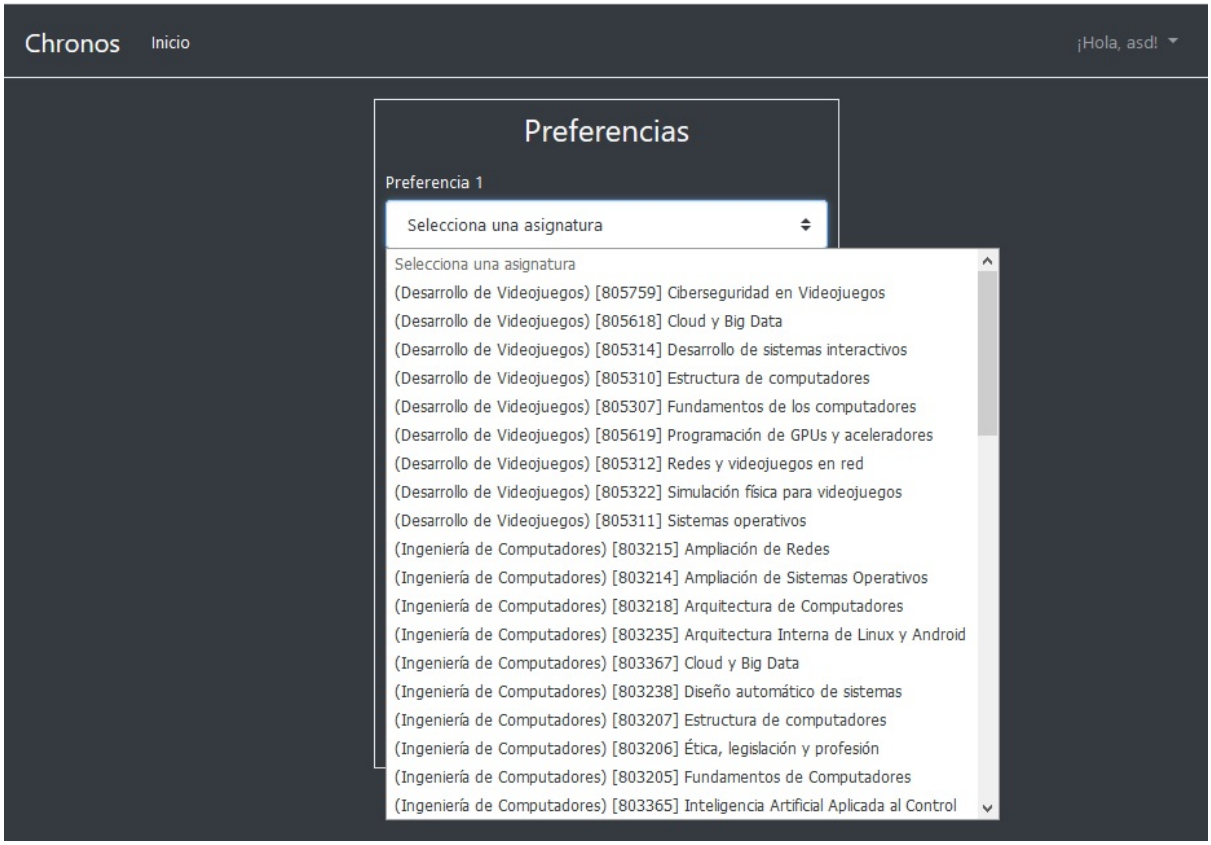


Figura 4.13: Selección asignatura preferencia

El docente seleccionará sus preferencias y las podrá guardar. Una vez que las guarde verá

lo siguiente:

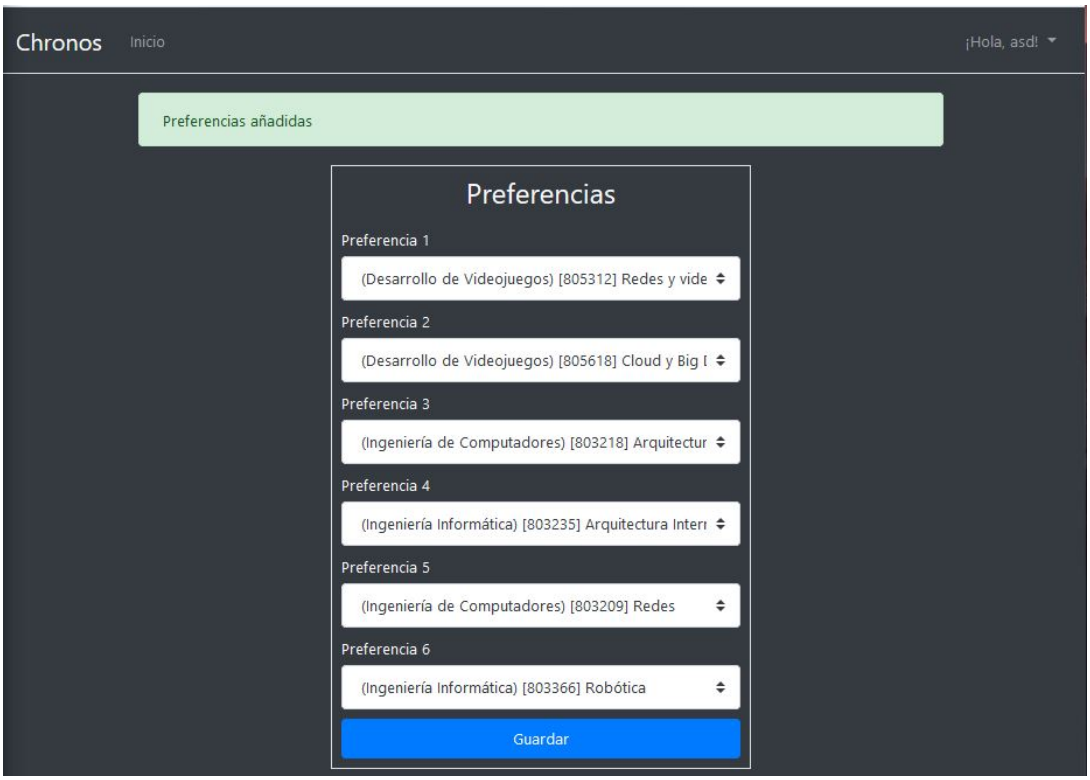


Figura 4.14: Preferencias añadidas

El docente es capaz de cambiar su propia contraseña. Tendrá que introducir la actual y dos veces la nueva.

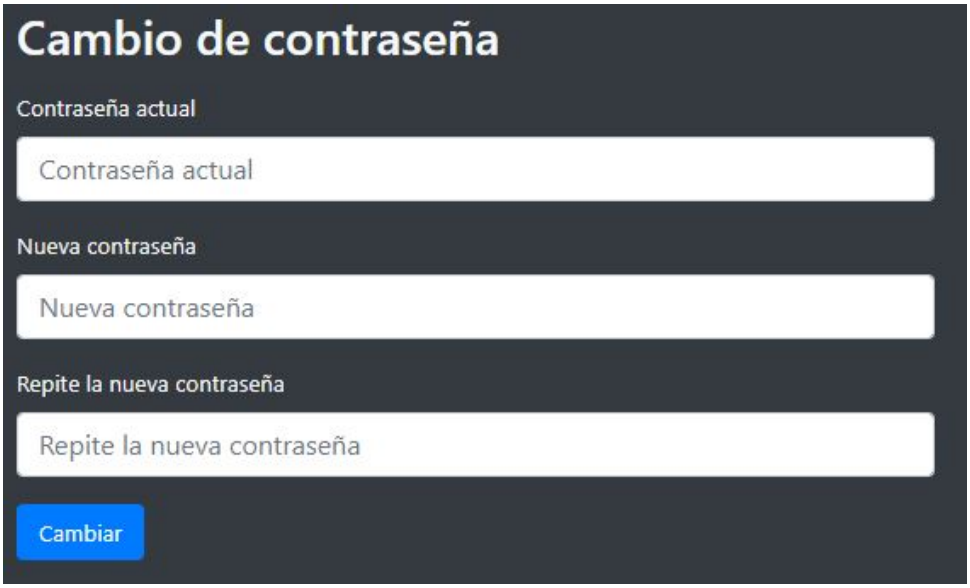


Figura 4.15: Cambio de contraseña en Docente y Gestor

Esta opción es común con Gestor.

4.4.4. Gestores

Finalmente, vamos a ver cómo los gestores pueden acceder al horario. Al igual que los docentes, tendrán que acceder con usuario y contraseña:

Figura 4.16: Inicio gestión

Una vez han realizado el registro, los gestores verán el siguiente inicio:

Figura 4.17: Pantalla inicio gestores

El gestor podrá modificar las facultades, las carreras, los itinerarios, los departamentos, las asignaturas, las clases y los docentes. A continuación vamos a ver cómo modificar cada uno de los anteriores.
Primero vamos a ver cómo modificar las facultades:

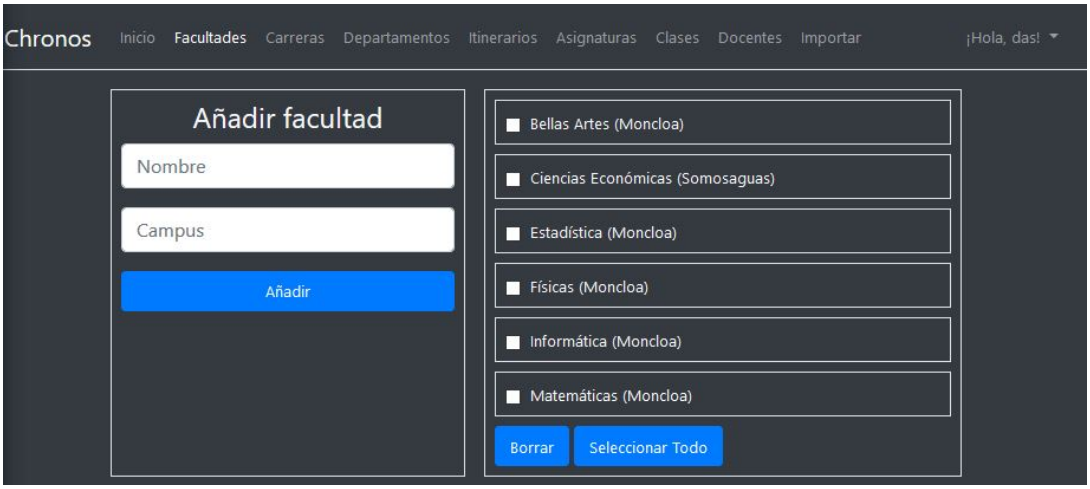


Figura 4.18: Gestionar facultades

Como podemos ver, a la izquierda podremos añadir las distintas facultades con su nombre y al campus al que pertenecen, y a la derecha, podremos ver las facultades añadidas.
Seguidamente, vamos a ver cómo modificar las distintas carreras:

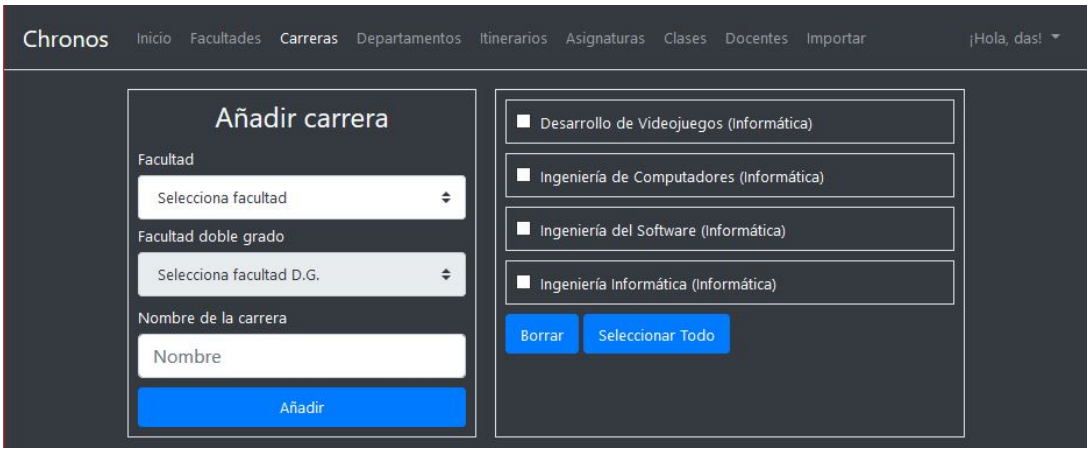


Figura 4.19: Gestionar carreras

En la columna de la izquierda podremos crear las distintas carreras y en la columna de la derecha nos mostrarán las distintas carreras añadidas.
Posteriormente, vamos a ver cómo modificar el departamento:

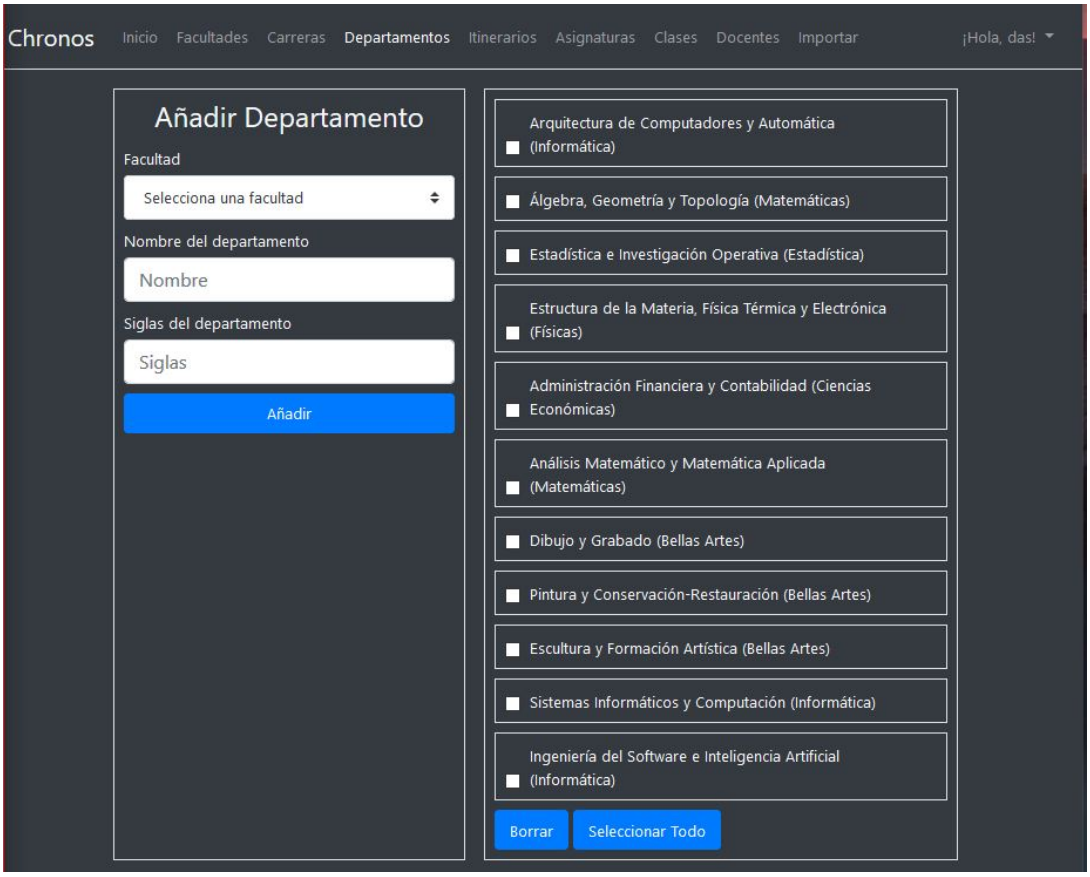


Figura 4.20: Gestionar departamento

En la columna de la izquierda podremos añadir el departamento de cada facultad y en la columna de la derecha podremos ver los distintos departamentos añadidos.
A continuación, vamos a ver cómo modificar el itinerario:

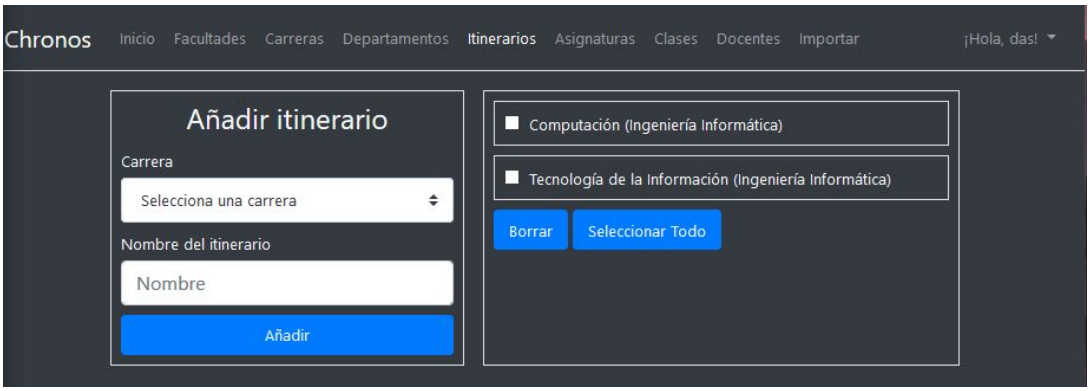


Figura 4.21: Gestionar itinerario

En la columna de la izquierda se selecciona la carrera con el nombre del itinerario y en la columna de la derecha vemos los itinerarios añadidos.
Seguidamente, vamos a ver cómo gestionar las asignaturas:

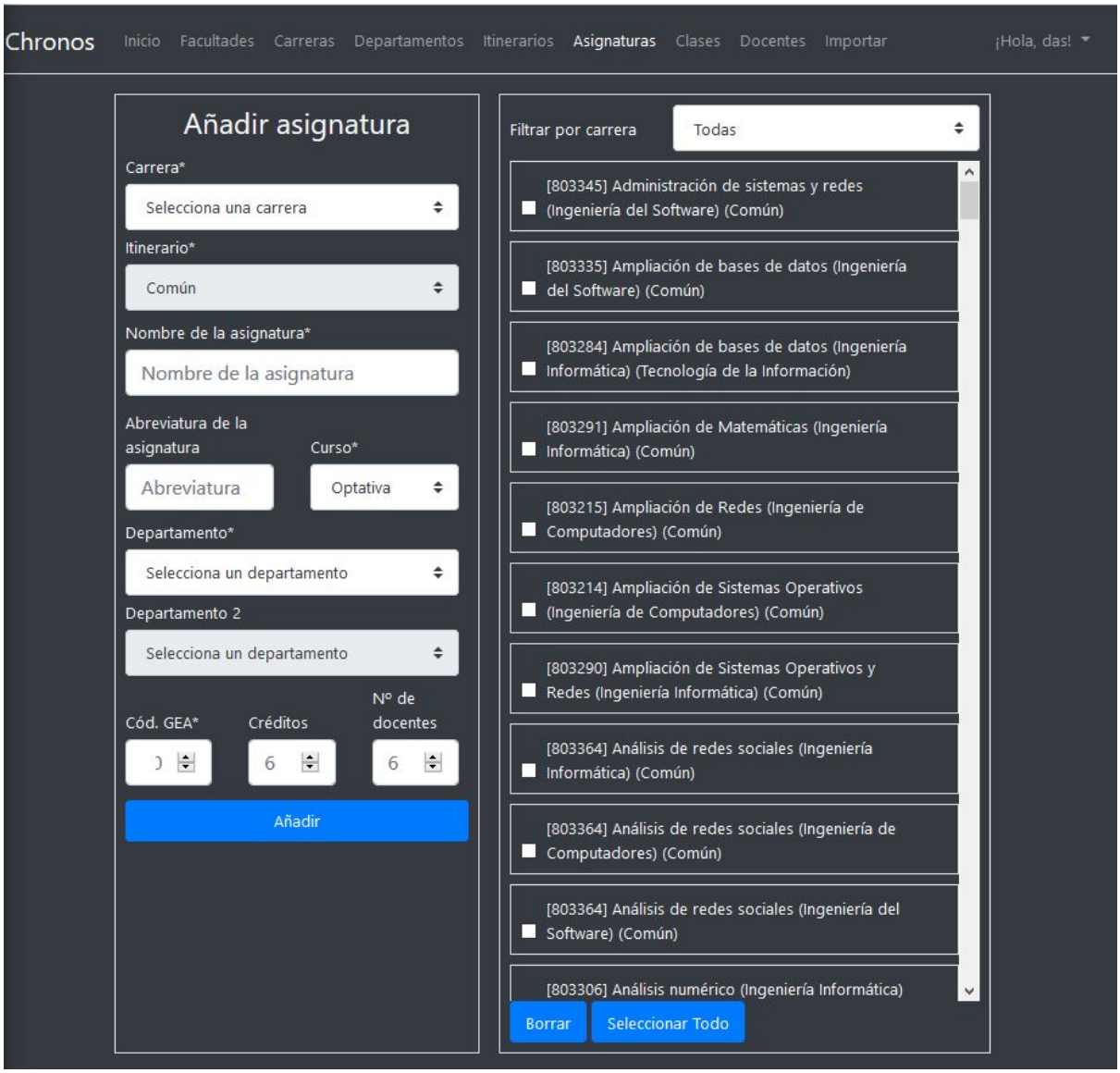


Figura 4.22: Gestionar asignatura

En la columna de la izquierda podremos añadir la asignatura, para ello, seleccionaremos la carrera, el itinerario, el nombre de la asignatura, la abreviatura de la asignatura, el curso, los departamentos, el código de GEA, créditos y números de docentes, y en la columna de la derecha veremos todas las asignaturas añadidas.
Seguidamente, vamos a ver cómo gestionar las clases:

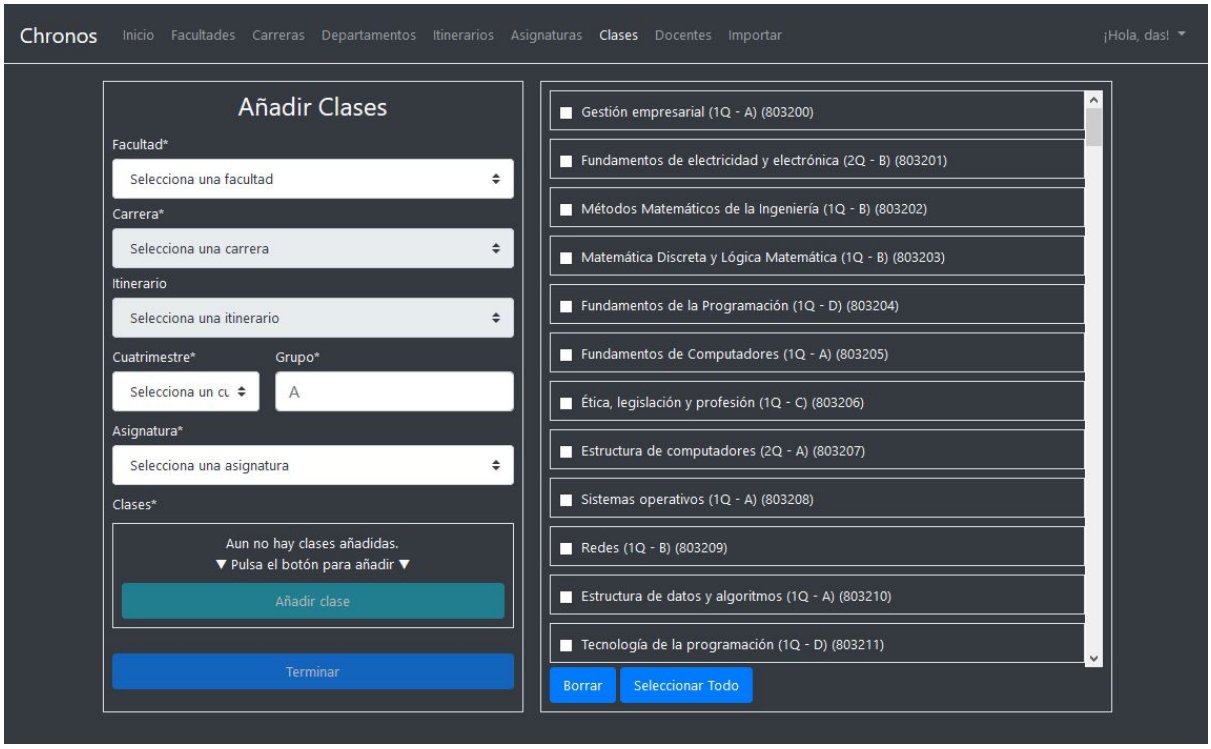


Figura 4.23: Gestionar clases. Paso 1

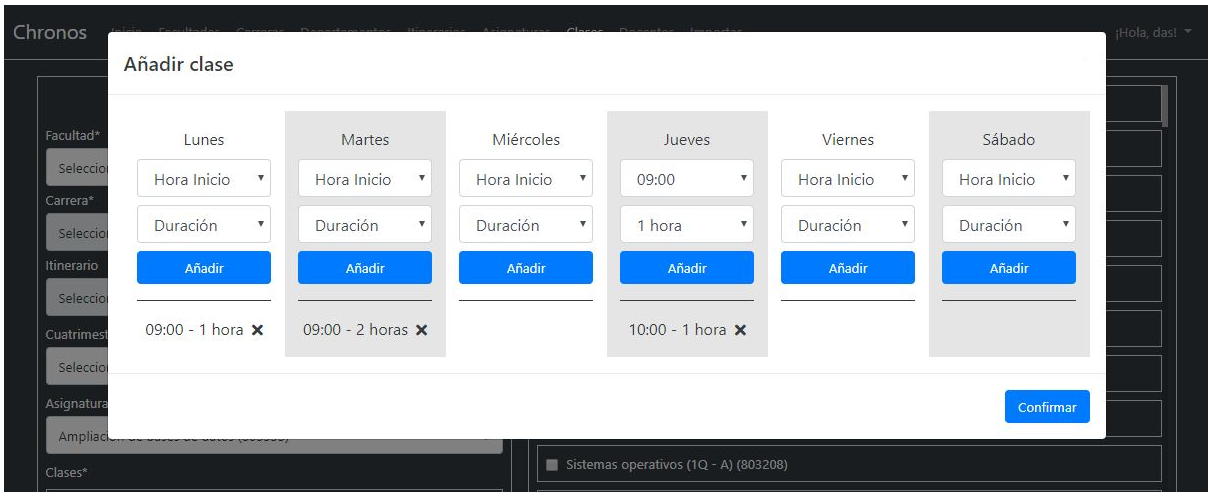


Figura 4.24: Gestionar clases. Paso 2

En la columna de la izquierda podremos añadir la clase, para ello primero seleccionaremos la facultad, después la carrera, seguidamente el itinerario, a continuación el cuatrimestre, y por último, grupo y asignatura, y en la columna de la derecha veremos las clases añadidas.
Tras seleccionar la asignatura, indicaremos la hora de inicio y la duración de cada clase. Se pueden añadir varias clases el mismo día.
Seguidamente, vamos a ver cómo añadir docentes:

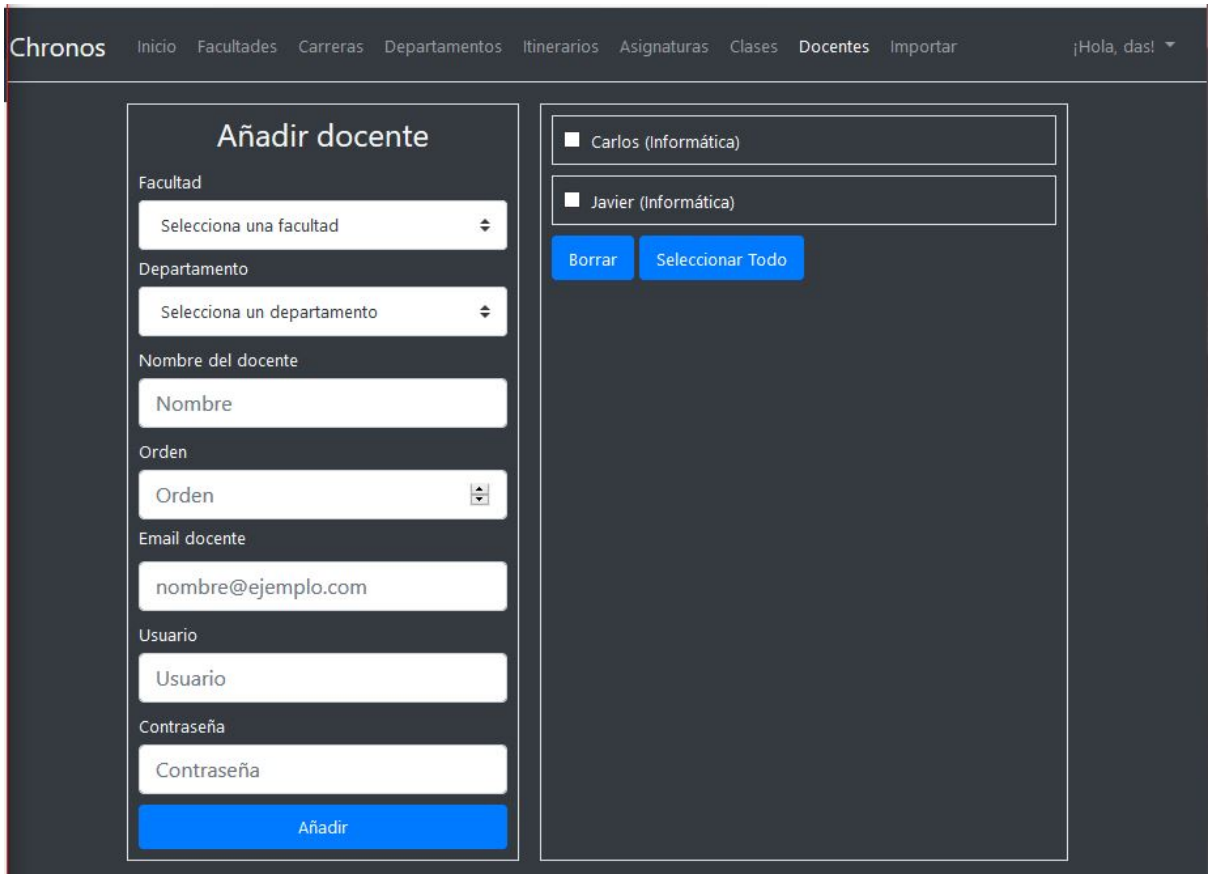


Figura 4.25: Gestionar docentes

En la columna de la izquierda podremos añadir los docentes, seleccionando primero la facultad, después el departamento, añadiremos el nombre del docente, el orden que llevan, el correo electrónico y le daremos su nombre de usuario y contraseña, y en la columna de la derecha veremos a los profesores añadidos.
Finalmente, podremos importar tanto los horarios como el reparto docente en formato .csv:

Figura 4.26: Importar

En la columna de la izquierda podremos importar los horarios y en la columna de la derecha podremos importar los horarios.

4.5. Gestión de riesgos activa

A lo largo de esta aplicación y del desarrollo de la misma hemos podido comprobar unos cuantos errores, no de la aplicación, y los hemos recogido en esta parte de la documentación como gestión de riesgos activa. Se va a proceder a describir cada uno de los riesgos encontrados junto a una posible solución o si se acaba asumiendo.

4.5.1. Riesgos del planteamiento del proyecto

- Riesgos de bajas en el equipo: Al ser un proyecto horizontal y haberlo creado con el sistema Kanban cada uno de los integrantes del proyecto era indispensable, si se hubiera producido una baja en el equipo habría conllevado la desestructuración completa del Trabajo de Fin de Grado.

- Aprendizaje de tecnologías usadas: Uno de los miembros del equipo controlaba todas las tecnologías a usar, por lo que parte de la documentación fue aprendizaje por el resto de los miembros del grupo de estas tecnologías. En el supuesto caso de que no se hubiera previsto ese riesgo, habría que haber cambiado la tecnología completa para adaptarla a conocimientos de ambos integrantes.

- Gestión del tiempo total: Al ser una aplicación online con una carga del servidor se han tenido que conseguir varios hosting gratuitos para realizar pruebas. Siempre con la tensión de que podría ser el último hosting gratuito que se encontrara, y por lo tanto, se iba contrarreloj para realizar algunas pruebas con usuarios. Esto se habría solucionado con el uso de XAMPP poniendo el servidor en local, pero para las pruebas era mejor tenerlo todo online.

4.5.2. Riesgos de la creación de la aplicación

- Al tener pocas herramientas para poder depurar lenguaje de marcas cada paso que se daba con conexión a servidor y a base de datos generaba mucha tensión por si nos íbamos a cargar algo del proyecto.

- Problemas de escalabilidad: Realizar un proyecto web conlleva muchísimo trabajo, que no es visible, pues lo que acabamos viendo es el frontend, por lo que el diseño tendría que ser perfecto, para hacer justicia la trabajo realizado. Esto hizo que tuviéramos que aprender mucho de CSS.

4.5.3. Riesgos con la aplicación finalizada

- Riesgo del estándar: Una vez creada la herramienta para que funcione para siempre se tiene que convertir en un estándar, por lo que, habría que implantarlo y hacerle publicidad para que todas las facultades supieran de su existencia.

- Riesgo de fallos por base de datos: Actualmente la aplicación funciona y es escalable para todos los datos obtenidos, pero sin un correcto mantenimiento puede

que en algún futuro de algún fallo relacionado con alguna asignatura nueva o con contingencias entre asignaturas.

- Riesgos de caída del servicio por gran cantidad de conexiones simultáneas: Cuando la aplicación tendría un mayor número de visitas sería en los días previos a los periodos de matriculación. Es importante garantizar el servicio, sobre todo, en esas fechas.

Capítulo 5

Arquitectura e implementación

Diagramas explicativos de la herramienta y explicación de como se forma la herramienta completa.

En este capítulo se tratará:

5.1. Servidor

En el curso 2017-2018, la UCM tenía un total de 71.806 alumnos[17]. Debido a que no todas las personas tienen que matricularse el mismo día, y que no tienen el mismo día a día, estimamos que en torno al 35 % del total estaría haciendo uso de la aplicación simultáneamente, es decir, 24.000 personas aproximadamente. Es por ello que hay que tener en cuenta las características y la configuración del servidor en el que se aloja la aplicación.
Hemos realizado pruebas de estrés en un servidor virtual «WP Gestionado Pro» de IONOS. La máquina real cuenta con dos «Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz» (40 cores en total) y 125GB de RAM. Se realizaron unas pruebas de estrés en las que hasta 200 usuarios enviaban una petición pesada, procesar el horario de 60 créditos, cada diez segundos, durante cinco minutos. Esta es la gráfica resultante.

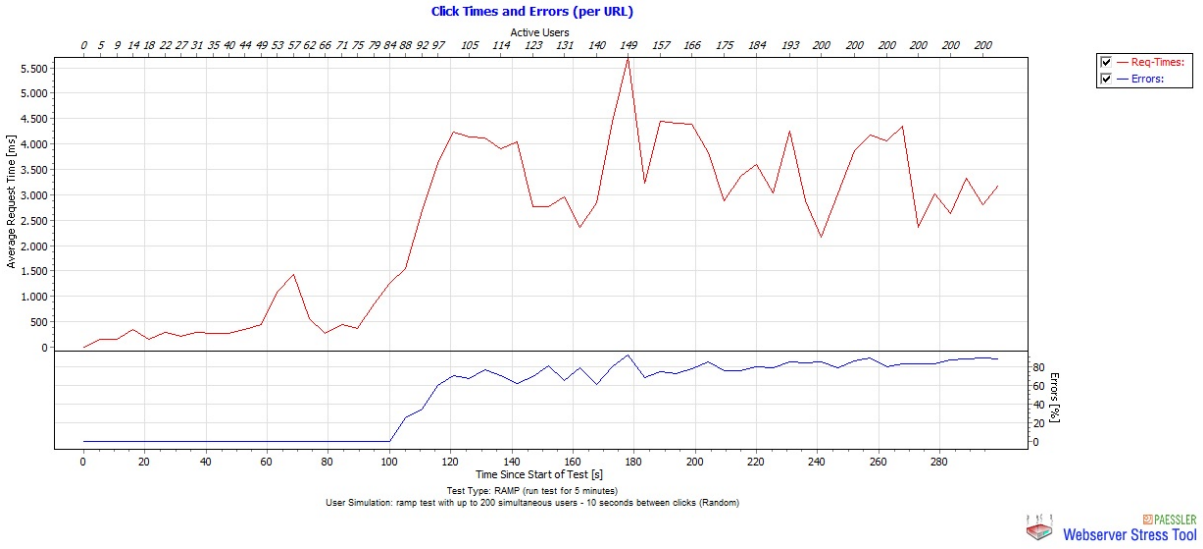


Figura 5.1: Relación entre número de usuarios activos, tiempo de petición y errores en el servidor de pruebas. Fuente: phpsysinfo[18]

No hemos podido averiguar de cuánta potencia podíamos hacer uso, pero estaba claro que de toda la que hemos indicado antes no.
A pesar de que su precio es de 15€/mes, bastante asequible, los errores empiezan a aparecer al alcanzar los 84 usuarios simultáneos. En caso de que se pudiera contratar varios de estos servidores y conectarlos a un balanceador de carga, caso que no es posible porque el proveedor no lo permite, sería necesario invertir unos 4.300€/mes, 516.000€/año, para que se garantizase el funcionamiento correcto de la aplicación hasta en los momentos de máxima afluencia de usuarios.
La aplicación está estructurada de tal forma que soporta la escalabilidad e incluso la ejecución en contenedores como Docker o Kubernetes. Es por esto que una de las mejores opciones sería optar por servicios como los de Amazon Web Services (AWS)[19], que permiten reducir el gasto en infraestructura al solo pagar por la potencia que se utilice. Así, durante el periodo en el que apenas se reciban visitas, se puede tener contratada una potencia capaz de soportar entre 50 y 250 usuarios simultáneos y, con ello, ahorrar una cantidad significativa de dinero.

5.1.1. Escenarios posibles utilizando AWS

Lo primero en lo que pensamos fue en que la aplicación podría estar disponible cualquier día, a cualquier hora. Por ello, nos dividimos el año en dos tipos de días: los «días valle», días en lo que el número de usuarios es mínimo, casi nulo; y los «días punta», los días previos y pertenecientes al periodo de matriculación. Fijamos el número de días punta en 40, 20 por cada periodo de matriculación. El resto serían días valle, 325 días.
Para que la aplicación funcione, sería necesario utilizar los servicios de Amazon EC2 [20], para tener un servidor; de Amazon EBS[21], para tener almacenamiento; y de Amazon S3 [22], para tener una base de datos.
Estos serían los resúmenes económicos dependiendo de si la aplicación fuera accesible en todo momento o solo durante unos días.

5.1.2. Aplicación accesible en todo momento

Como no será necesaria mucha potencia durante los días valle, se optó por planear la contratación de una instancia «t3.small».
Para los días punta se decidió que la instancia «t2.2xlarge» sería más que suficiente para soportar todas la carga de trabajo.
Se dispone de todo el almacenamiento y bases de datos que se quiera y se paga por lo que realmente se utiliza.
Con estos datos en mente, había que calcular costes.

- **t3.small** sería utilizado durante 325 días tiene un coste total de **164,88€/año**.
- **t2.2xlarge** sería utilizado durante 40 días tiene un coste total de **405,51€/año**.
- El **almacenamiento** utilizado no llegaría a 1GB, cuyo coste es **1,5€/año**.
- La **base de datos** tiene un coste por almacenar los datos y otro coste por acceder a ellos. En total el gasto de esta funcionalidad es **3,5€/año**.

En total, el coste sería de **575,39€/año**.

5.1.3. Aplicación accesible por tiempo limitado

Manteniendo el planteamiento inicial, la aplicación solo podría utilizarse 40 días al año. Si se contrata todo lo anterior, salvo «t3.small», el coste sería el siguiente

- **t2.2xlarge** sería utilizado durante 40 días tiene un coste total de **405,51€/año**.
- El **almacenamiento** utilizado no llegaría a 1GB, cuyo coste es **1,5€/año**.
- La **base de datos** tiene un coste por almacenar los datos y otro coste por acceder a ellos. En total el gasto de esta funcionalidad es **3,5€/año**.

En total, el coste sería de **410,51€/año**.

5.2. Capas de la aplicación

5.2.1. Backend

A pesar de no ser lo visual, aquello que perciben los usuarios, el contenido animado y colorido, la aplicación no existiría de no ser por el backend.
Antes de comenzar a desarrollarlo, estuvimos estudiando cuál era la mejor manera de estructurarlo, cómo se organizaría la información de manera interna y cómo se tratarían los datos para enviarlos tanto a la base de datos como al frontend.
Lo primero en lo que nos pusimos a trabajar fue la base de datos porque sería en ella donde se apoyaría el resto del proyecto. Necesitábamos diseñar una base de datos que fuera sencilla a la par de completa. Esto complicaría las cosas menos en caso de tener que escalar el proyecto. Fueron necesarias las ocho tablas que se muestran a continuación:

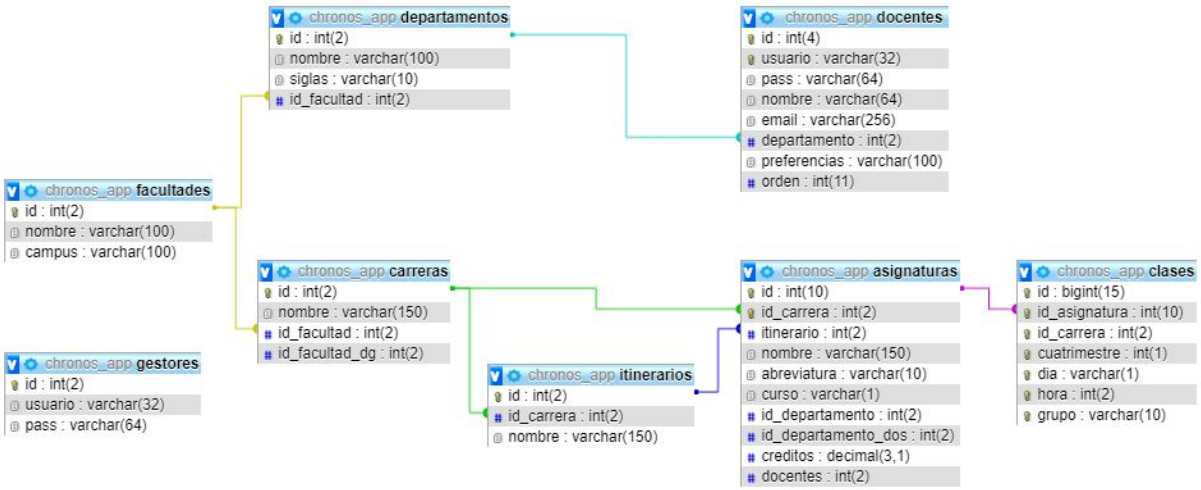


Figura 5.2: Tablas de la base de datos y claves foráneas

Una vez creada la base de datos, procedimos a preparar la estructura de los datos. Creamos tantas clases como tablas y un DAO para cada una de ellas. De esta manera, las clases tan solo se encargaban de almacenar la información y los DAO de insertarla y extraerla de la base de datos.
Preparada toda la estructura de datos, podíamos empezar a trabajar en los algoritmos: el de procesado de horarios, el de reparto docente y los de importación de datos mediante ficheros.
Para facilitar la navegación a los usuarios que vuelven, preparamos una cookie de sesión en PHP para que recordara algunos datos en función de su rol:

- **Estudiante**: la carrera y el itinerario.
- **Docente**: ID, nombre del usuario, departamento y preferencias.
- **Gestor**: ID y nombre del usuario.

La gran mayoría de las llamadas que recibe el backend son asíncronas, es decir, que no se realizan en un salto de página y el usuario no es consciente de que se están realizando.

Para poder mantener el código organizado, decidimos concentrar todas las conexiones en un solo punto: async.php. Este fichero es el encargado de recibirlas, realizar la tarea que se solicita en ellas y devolver por la misma vía la información resultante de la acción. A este fichero no se puede acceder haciendo uso de una navegación normal. Si se intenta acceder, se es redireccionado a la página principal.
Salvo en el paso de la página principal al asistente, donde se aprovecha para generar la cookie de sesión, las conexiones asíncronas son las encargadas de hacer que los usuarios con el rol de Estudiante sean capaces añadir asignaturas a su listado y de recibir el horario una vez solicitan el procesado.
En los apartados de Docente y Gestor también se utilizan. Sobre todo, en el filtrado de las opciones de los menús desplegables. Seleccionar una opción en uno de estos menús puede condicionar el contenido del resto del formulario. Para saber qué opciones son las que hay que mostrar se hace uso de conexiones asíncronas.
Apenas habría que realizar modificaciones en caso de que se quisiera implementar una API Rest para conectar alguna aplicación externa. En esencia, esta parte del backend se comporta de manera muy similar.

5.2.2. Frontend

Para desarrollar la parte frontal, nos apoyamos en Bootstrap, un framework que ayuda a reducir considerablemente el tiempo a invertir en desarrollo de interfaces web.
Ya definidos los roles de usuario, se decidió separar el código Javascript de manera que cada apartado perteneciente a cada rol cargara solo el fichero correspondiente. Al no haber tanto código CSS propio como para ser necesario separarlo en ficheros como el Javascript, se dejó todo el un solo fichero.
Se apostó por una paleta de colores oscura en casi todos los elementos para que agresivo a la vista.

- **Estudiante**.
  La página de inicio consiste en un formulario en el que se nos pregunta primero la carrera y después el itinerario, en caso de que existiera. Una vez indicados los datos, se nos llevará a la pantalla del asistente. El código Javascript de la portada se centra en el control del formulario.
  El asistente tiene su propio fichero con código Javascript. Las funciones que hay en este se centran en el control del formulario encargado de añadir asignaturas y en almacenar las clases para mostrar el horario construido a partir de las asignaturas que ha indicado el usuario.
  Gracias al uso de jQuery-ui, se pueden ordenar las asignaturas seleccionadas para indicarle la prioridad a la aplicación con tan solo arrastrar y soltar con el ratón.
  El el asistente se compone de dos columnas: una para las opciones de «Añadir asignatura», «Vaciar horario» y «Procesar horario» y el listado de asignaturas seleccionadas; y otra, que ocupa casi toda la pantalla, que contiene el horario de un cuatrimestre y un botón para alternar entre el primero y el segundo.

- **Docente**.
  Se mostrará un formulario de inicio de sesión con un diseño similar al del formulario de los Estudiantes.
  Una vez dentro, se les ofrecerá la posibilidad de establecer sus preferencias, de

cambiar la contraseña y de cerrar sesión.
En este caso, se ha apostado por un diseño de una sola columna centrada.

- **Gestor**.
  Se mostrará un formulario similar al de Docente. Tras iniciar sesión, se mostrará un menú superior en el que se mostrarán todas las cosas que puede gestionar y, en filas de 3 elementos, se muestran pequeños recuadros en los que se muestra cuántas entradas hay en cada tabla de la base de datos.
  Cada formulario de gestión tendrá filtrados mediante AJAX para facilitar los procesos de rellenado de formularios.

5.3. Flujo de ejecución

- **Estudiante**.
  Usamos POST clásico para cambiar la vista de selección de carrera a la de selección de horarios, cargando la asignaturas posibles.
  También usamos AJAX para mandar mensajes desde el usuario a la aplicación, mandando información de qué datos recopilar, o usar, al controlador.
  En concreto los datos de las asignaturas de las que quiere matricularse y su disponibilidad horaria. Usamos estos datos para construir el horario, modificando la vista de forma asíncrona. Gracias a esto evitamos refrescos, haciendo la interacción para el usuario más cómodo, agradable y fluida.

- **Docente**.
  Utilizamos POST para iniciar sesión, recopilando y mostrando la información si el login es correcto, y en caso negativo, mostrando nuevamente el formulario de inicio de sesión. También se utiliza POST para guardar los cambios en las asignaturas que son guardadas como preferentes.
  Se utiliza AJAX para cambiar el contenido de los menús desplegables.

- **Gestor**.
  Usamos POST para el login, obteniendo la información del gestor, recopilándola y mostrándole su vista correspondiente si el login es valido. Si no, mostraremos el login nuevamente. También se usa POST para añadir, editar o eliminar elementos de la base de datos.
  AJAX, de igual forma que en Docentes, lo usamos para realizar filtrados en los desplegables de los formularios.

Capítulo 6

Resultados

Discusión sobre los resultados finales del proyecto.

En este capítulo se tratará:

6.1. Discusión de resultados

Para poder realizar el proyecto se ha organizado un gran estudio de herramientas que podían hacer exactamente lo mismo que la nuestra. Al no encontrarla, se ha procedido al desarrollo de la solución. Una vez realizadas pruebas con tres tipos de usuarios: estudiante, docente y gestor. Se puede decir que la aplicación es enteramente funcional y está lista para ser probada sin ningún tipo de problema. La aplicación tiene las siguientes características:

- **Compatible**. La aplicación se puede utilizar tanto en los navegadores actualizados como en los que están unas cuantas versiones atrás.
- **Sin colores agresivos**. Utilizamos lo máximo posible una gama de colores que daña lo mínimo posible a la vista.
- **Ligero**. La aplicación completa apenas supera los seis megabytes.
- **Fácilmente instalable**. Tan solo se necesita que el servidor tenga una base datos, o permitir la conexión a una externa, y un intérprete de PHP.

6.2. Problemas al proyecto

Dentro de los problemas encontrados en el proyecto, encontramos uno muy importante respecto al diseño web. Crear una aplicación web requiere muchísimo tiempo y muchísimas pruebas constantes, una gran cantidad de documentación, esto ocurre el riesgo de que sea menos vistosos que una aplicación creada con un lenguaje de programación corriente, por lo que es uno de los principales problemas que se han encontrado.
Otro gran problema que nos hemos encontrado a la hora de realizar pruebas para la aplicación era la falta de un estándar entre facultades que nos facilitara el trabajo. Mientras que en informática seguía un estándar muy bueno, en otras facultades casi tenías que mirar un tutorial para poder entender cómo se realizaban los horarios. A esto habría que añadirle que dichos horarios se publicaban en sus páginas web en formato PDF, lleno tablas y textos en colores diversos, lo que hace prácticamente imposible que se puedan indexar. Este ha sido el principal problema y posiblemente, lo que ha detenido el proyecto varias semanas.
Un problema típico que surge en los proyectos de este tipo de aplicaciones es la protección ante ataques. Necesitas un decompilador o similar para averiguar cómo funciona una aplicación de escritorio por dentro, pero una aplicación web está mucho más expuesta ya que, con tan solo presionar F12 en el explorador, se puede ver casi la mitad del código que hacer funcionar a la aplicación. Evitar los ataque de inyección SQL y XSS, entre otros, ha sido una de nuestras mayores prioridades.

6.3. Contribución personal

En esta sección hablaremos acerca de las contribuciones de cada uno de los integrantes del grupo, tanto las realizadas de forma conjunta como las realizadas individualmente.

6.3.1. Contribuciones realizadas por ambos

Las siguientes son las tareas en las que los dos hemos trabajado conjuntamente.

Búsqueda de herramientas similares

Lo primero que hicimos fue investigar si existía, o no, alguna herramienta que fuera igual, o similar, a la que teníamos planeado desarrollar. Encontramos muchas que ofrecían funcionalidades de calendario, pero solo una se acerca a lo que realmente buscábamos. Se trata del sistema de automatrícula, ofrecido por el portal GEA. En él, durante el proceso de matriculación, el alumno tiene que seleccionar las asignaturas en las que quiere matricularse y, a continuación, el grupo. Antes de finalizar el proceso de matriculación, la herramienta muestra al alumno cuál será su horario si confirma la selección. El problema que esto conlleva es que el tiempo corre en contra del estudiante. Cuanto más tiempo pasa, más probable es que se hayan ocupado todas las plazas de una asignatura.
Por otro lado, vimos que los horarios se publicaban unas semanas antes del periodo de matriculación. Esto permitía que los estudiantes pudieran planificar en qué asignaturas y en qué grupos matricularse antes de comenzar el proceso de matriculación. El inconveniente que había era que los estudiantes tenían que hacer combinaciones manualmente hasta llegar a la que satisfacía sus necesidades. Una tarea laboriosa que muchos deciden no hacer, prefiriendo necesitar más tiempo en el proceso de matriculación.

Obtención de la información

Nuestro proyecto necesitaba nutrirse de toda la información acerca de las asignaturas, sus grupos y sus clases. Estudiamos las formas por las cuales pudiéramos conseguir dicha información.
La primera fue crear un crawler que recorriera todas las páginas de los horarios. Este almacenaría toda la información para procesarla más adelante y poblar la base de datos. Mientras empezábamos a preparar el algoritmo de indexado, nuestro tutor nos comentó que existía un fichero de hojas de cálculo con la misma información y nos recomendó que intentásemos extraer la información de este. Dicho fichero contenía varias hojas con la información organizada en aulas y laboratorios. Se mostraba una tabla que abarcaba de lunes a viernes y de nueve de la mañana a nueve de la noche para cada uno de ellos. Algunos datos importantes como el grupo, el departamento o si la clase empezaba a las horas y media, en lugar de a en punto, se indicaban aplicándole formato a la celda o al contenido. Finalmente, esta forma de obtener los datos quedó descartada porque no encontramos ninguna librería que pudiera leer dicho formato.
En un intento de no volver a la idea del crawler, le preguntamos a nuestro tutor si la información del fichero que nos proporcionó se podía encontrar en algún formato más primitivo. A poder ser, en texto plano. Nos dijo que, muy probablemente, el vicedecano de ordenación académica de nuestra facultad, Fernando Rosa, podría proporcionarnos el fichero en un formato que nos pudiera resultar válido.
Tras poder reunirnos con él, nos hizo saber que ese fichero existía. En él, se indicaban los siguientes datos de las asignaturas: identificador único, curso, nombre, siglas, departamentos responsables, cuatrimestre y el horario de clases para cada grupo. Todo ello era convertible a formato CSV. El único inconveniente era que no se indicaban explícitamente los créditos de cada asignatura.

Estructura de la información

Hemos de admitir que la estructura nos resultó un quebradero que cabeza. Al inicio pensábamos que con cuatro o cinco clases, sus correspondientes DAO y su tabla en la base de datos, estaría todo más que preparado para empezar a programar. Al final resultaron ser ocho tablas en la base de datos, con todo lo que ello conllevaba.
Una vez ya confirmada la estructura, comenzamos a estudiar cómo guardar la información. Al final vimos que solo un dato tenía que ser codificado si queríamos ganar eficiencia: la hora. MySQL es capaz de almacenar horas de varias formas, el problema está en que manejar tiempos entre PHP y MySQL aumenta ligeramente los tiempos de ejecución. Es cierto que se aumentan muy poco pero, en el caso de que haya un gran número de usuarios utilizando la aplicación de manera simultánea, puede marcar la diferencia. Finalmente nos decidimos por convertir las horas en un número de hasta dos cifras. Algo muy fácil de manejar tanto en MySQL como en PHP. Dicho número representa el número de bloques de treinta minutos que han pasado desde las ocho de la mañana. Por ejemplo, las 10:00 se almacenaría como un 4 y las 18:30 como un 21. Decidimos que fueran bloques de media hora para que fuera compatible con las carreras cuyas asignaturas comienzan a las horas y media.

6.3.2. Contribuciones realizadas por Javier

Procedemos a enumerar las aportaciones realizadas por Javier.

Algoritmo importación de datos mediante fichero

Debido a mi trabajo, estoy acostumbrado a trabajar con bases de datos en ficheros de texto en casi cualquier formato. Es por ello que me hice cargo de esta tarea.
Después de analizar qué datos nos proporcionaba el fichero, lo primero que hice fue exportarlo en formato CSV, ya que hay más documentación sobre cómo leer ficheros en este formato que ficheros XLS. A continuación, comencé a preparar las expresiones regulares para obtener los datos de tal manera que resultara más cómodo y sencillo tratarlos. Dejando a un lado la lectura del fichero, el algoritmo lo dividí en dos partes: la encargada de insertar las asignaturas, y la encargada de insertar las clases. Una vez leída una línea, se separan los datos y se inserta la asignatura en la base de datos. Seguidamente, se insertan sus clases. Esto se hace así porque la base de datos utiliza claves foráneas para mantener la consistencia, por lo que no permite insertar clases si la asignatura a la que pertenecen no existe.
A pesar de que no se indicaba cuántos créditos valía una asignatura, estos se podían calcular a partir del número de horas de clase a la semana y de si la asignatura era cuatrimestral o anual. Una vez insertadas todas las clases, el algoritmo revisa cuántas horas de clase tiene cada asignatura y actualiza los créditos de esta en la base de datos.

Estudio de la situación

En cuanto un estudiante se quiere matricular de asignaturas que no pertenecen a un mismo curso, se enfrenta a que los horarios se solapen y tenga que estar en dos clases al mismo tiempo. Un problema que se complica conforme se van añadiendo asignaturas a la lista. Un problema que una persona puede tardar hasta horas en encontrar una solución, mientras que una computadora puede tardar unos pocos segundos en calcular todas las

soluciones válidas.
Observando, pude comprobar que todos los estudiantes se pueden organizar en tres grupos: los que tienen todo el día disponible para estudiar, los que solo tienen las mañanas disponibles y, por el contrario, los que solo pueden asistir a clases por la tarde.
Los docentes también tienen un problema relacionado con el horario, pero no está relacionado con la disponibilidad, sino con el orden de selección de las asignaturas. Nuestro tutor nos comentó que tienen una hoja de cálculo en la que, a partir de ciertos parámetros, se les asigna un número a cada docente. Dicho número es el puesto que este ocupa en el orden de selección de asignatura. Poco a poco, la hoja de cálculo se va rellenando hasta que el último docente realiza la última inserción.
Por ello, hemos diseñado un modelo más rápido, en el que los profesores indican hasta un máximo de seis asignaturas de las cuales quisieran ser docentes y de forma automática se realiza el reparto. El programa recibirá cuántas plazas tiene cada asignatura y el orden de selección de los profesores. Con estos datos, comenzará a asignar las plazas siempre que sea posible. En caso de no serlo, probará con las demás preferencias.

Vistas

Teniendo en mente que la aplicación está pensada para que pudiera llegar a ser utilizada por toda la comunidad académica, teníamos la obligación de hacer que fuera lo más intuitiva posible. Esto me llevó a hacer casi una decena de prototipos en papel. Todo sobre el papel funciona y parece que servirá, ya que no se guardan las proporciones. Para parar el ciclo, decidí buscar herramientas de diseño de interfaces. Conocía Balsamiq, pero el resultado no me parecía lo suficientemente profesional. Las líneas que parecían pintadas a mano y todos los textos escritos en Comic Sans hicieron que descartara esta opción. Al poco de seguir buscando encontré una de las pocas aplicaciones de Adobe que son gratuitas y que cumplía con las exigencias: Adobe Xd [23]. Utilicé esta herramienta para diseñar completamente todas las vistas correspondientes al rol de Estudiante y la estructura básica de las páginas de Docente y Gestor.
Para agilizar el proceso de desarrollo de las interfaces, me apoyé en el CSS y el Javascript que proporciona Bootstrap. Además de usar el jQuery que trae este, hicimos uso de jQuery-ui, una librería específica centrada en hacer que las interfaces de usuario sean más amigables. Todo esto, junto con al HTML, CSS y Javascript que desarrollé, forman las vistas que verán los futuros usuario.

Conexiones asíncronas

Para controlar la información que fluirá de fuera hacia adentro de la aplicación, decidí que lo mejor sería utilizar un intermediario, un controlador. Todas las peticiones asíncronas que se realizan pasan a través de este elemento. Así, se disminuye el número de puntos de entrada de posibles atacantes.

6.3.3. Contribuciones realizadas por Carlos

Las contribuciones realizadas por Carlos son las siguientes:

Estudio de roles de usuario

Al ser una aplicación destinada a estudiantes y profesores, esta tarea la comenzamos sabiendo que, como mínimo, tendríamos estos dos roles. Seguidamente nos surgió la pregunta «¿Quién se encarga de que la información esté en el sistema?». La respuesta fue el tercer y último rol: el gestor.
A continuación, entraremos más en detalle de cada uno de los roles que se puede tener:

- **Estudiantes**. Para comenzar a utilizar la aplicación con este rol, tan solo hay que entrar a la página principal. Indicamos qué carrera y qué itinerario, en caso de que existan, estamos estudiando o estamos interesados estudiar y ya podemos comenzar a usar la aplicación. Añadiremos las asignaturas que nos interesen, indicaremos nuestra disponibilidad (día completo, mañana o tarde) y le indicamos a la aplicación que queremos que nos genere el horario.

- **Docentes**. Los docentes tendrán su propio apartado privado dentro de la aplicación, al cual accederán mediante usuario y contraseña. Para ello necesitarán que el gestor les registre. Una vez dentro, podrán cambiar la contraseña e indicar cuáles son las asignaturas de las cuales quieren dar clase. Dichas preferencias podrán cambiarse tantas veces como quieran hasta que comience el reparto. Una vez comienza dicho proceso, las preferencias se bloquean para evitar errores.

- **Gestor**. Este usuario tiene acceso a toda la información contenida en la aplicación salvo a las preferencias de los docentes. Es capaz de crear, modificar y eliminar cualquier elemento, ya sea facultad, carrera, itinerario, departamento, asignatura, clase o docente. También es responsabilidad suya importar los ficheros de horario y de reparto docente.

Algoritmos

Primero busqué ideas en internet de como solucionaron este problema, estas incluían búsqueda en espacios o algoritmos genéticos. Debido al alto consumo en memoria, que nos interesa una solución válida, más que comparar cuál es mejor; y con la idea de que el programa sea aplicable a tecnología móvil, me decanté por un algoritmo de backtracking que minimizase el coste en espacio sin aumentar el coste de tiempo, o aumentándolo levemente.
Tras eso, me planteé qué variables necesitaba y cuáles podía pasar por referencia, y cuáles no, para mantener costes. Me quedé con dos atributos para los horarios, uno para el primer cuatrimestre y otro para el segundo. Ambos arrays de cinco por veinticuatro, al tener un tamaño constante, recorrerlos es constante, y solo los modifico, nunca los copio, aumentando la eficiencia. Y un array adicional que guarda la información suministrada por el usuario; este array tiene un tamaño de (n*k), siendo n las asignaturas que desea matricularse el alumno, y k los grupos de esa asignatura (en torno a 8 en las que más). Nuevamente este array una vez generado, no se vuelve a modificar, ni copiar. Y para el algoritmo en sí necesité un número entero para marcar la profundidad, una estructura para guardar la solución (coste en espacio n), y un booleano para determinar si tenemos una solución válida o no.
Al ir programando, mi mayor problema fue la ausencia de declaración de tipos en PHP, que me obligaba a estar comprobando qué contenía las variables en todo momento, y que estuviesen bien usadas. El otro problema es el ya mencionado del espacio, lo que me

obligaba usar una cantidad limitada de variables, lo que se tradujo en tener métodos que deshiciesen pasos para mantener el control de los atributos, de modo que los datos no se corrompieran al ir entrando más profundamente en la recursión.
El de reparto docente me resultó mucho más fácil, ya que tan solo era recorrer el listado ordenado de profesores e ir probando si se podían apuntar en una asignatura o no.

Capítulo 7

Conclusiones y trabajo futuro

Resultados finales del trabajo, valoración por parte del equipo y proyecto de futuro.

En este capítulo se tratará:

7.1. Conclusiones

Este Trabajo de Fin de Grado se ha afrontado con la motivación de aprender, estudiar e investigar acerca de todo lo posible de cómo crear una aplicación para mejorar la gestión de horarios.
Hemos realizado numerosas investigaciones sobre aplicaciones similares a la que teníamos nosotros en mente, para ello nos las hemos instalado y hemos visto cómo poder adaptarlas a los horarios de tanto estudiantes como docentes. Pero hemos visto que no nos permiten gestionar de una manera óptima nuestros horarios.
En lo referente al nivel tecnológico, hemos aprendido como implementar una aplicación web rápida a pesar de utilizar algoritmos complejos.
Por lo que concluimos, nos gustaría poder llegar a implantarlo en todas las facultades, ya que facilitan, tanto a estudiantes como a docentes, la organización de los horarios. Para ello, nos gustaría poder contar con todos los horarios de todas las facultades para poder implementarlo.

7.2. Trabajo futuro

Dentro del trabajo que nos gustaría poder completar en un futuro está:

- La posibilidad de contingencias: Muchos alumnos por temas de horario o falta de plaza en algunas clases tienen que coger unas asignaturas a la misma hora que otras. Esto nos gustaría poder agregarlo en un futuro.
- Sistema de puntuación para estudiantes: De esta forma, se podría catalogar las asignaturas que son más fáciles y más difíciles, y las asignaturas que más gustan o menos gustan a los estudiantes.
- Sistema de recomendación de asignatura por puntos: Puede que sea una idea bastante complicada, pero estaría bien un autogenerador de calendario que le pongas unas características como por ejemplo, una restricción a las horas en las que puedes ir a clase, cuántos créditos quieres cursar, las dificultades de las asignaturas y que automáticamente te genere un horario.

Chapter 7

Conclusions and future work

Final results of the work, assessment by the team and future project.

In this chapter we will talk about:

7.1. Conclusions

This Final Degree Project has been addressed with the motivation to learn, study and investigate everything possible about how to create an application to improve the management of schedules.
We have done a lot of research on applications similar to what we had in mind, for this we have installed them and we have seen how to adapt them to the schedules of both students and teachers. But we have seen that they do not allow us to manage our schedules in an optimal way.
Regarding the technological level, we have learned how to implement a fast web application despite of using complex algorithms.
So we conclude, we would like to be able to implement it in all the faculties, since, they facilitate both the student and the teacher the organization of the schedules. For this, we would like to be able to count on all the schedules of all the faculties to be able to implement it.

7.2. Future Work

Within the work that we would like to be able to complete in the future is:

- The possibility of contingencies: Many students due to schedule issues or lack of place in some classes have to take some subjects at the same time as others. We would like to add this in the future.
- Scoring system for students: In this way, you could classify the subjects that are easier and more difficult, and the subjects that students like or least like.
- System of recommendation of subject by points: It may be a rather complicated idea, but it would be good to have a self-generated calendar that you put some characteristics such as a restriction to the hours you can go to class, how many credits you want to take, the difficulties of the subjects and that automatically generates a schedule.

Anexo A

Repercusión

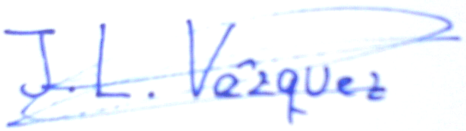
Chronos fue planteado como un proyecto de software libre para que se pudieran beneficiar de él no solo la UCM, sino cualquier persona, institución u organización que lo necesitase.
Esto llegó a oídos de «OTEA» [24], la Oficina de Software Libre y Tecnologías Abiertas de la UCM, la cual mostró su apoyo e interés desde el primer instante. Como muestra de ello, redactaron una carta de interés recomendando este proyecto porque, como dice en la misiva, "responde a una necesidad existente".
A día de hoy, Chronos se puede encontrar en el catálogo «Alejandría» [25], uno de los apartados del sitio web de dicha oficina.

OFICINA DE **SOFTWARE LIBRE**
VICERRECTORADO DE TECNOLOGÍAS DE LA INFORMACIÓN
UNIVERSIDAD **COMPLUTENSE** MADRID

Como director de la Oficina de Software Libre y Tecnologías Abiertas de la Universidad Complutense de Madrid, quisiera recomendar el Trabajo Fin de Grado “Chronos: sistema de simulación de horarios”.

Este TFG responde a una necesidad existente en la Universidad y supone una propuesta alineada con los esfuerzos de la Oficina.

Madrid, 14 de mayo 2019,



José Luis Vázquez-Poletti
Director de la Oficina de Software Libre y Tecnologías Abiertas (OTEA)
Universidad Complutense de Madrid, Spain
Web: https://www.ucm.es/oficina-de-software-libre/
E-mail: otea@ucm.es
Teléfono: +34913947600

Índice de figuras

Bibliografía

[1] https://horarios.coldideas.com/. coldideas.

[2] https://www.canva.com/design/DADaTCDBdh8/LhLtF7afeLB0Pe66fsOIew/edit?category=tACZCjS-Td0&utm_source=onboardinghttps://empresearandom.woffu.com/#/dashboard/admin. Canva.

[3] https://www.asctimetables.com/timetables_es.html#!/home/features. asctimetable.

[4] http://www.timetableweb.com/es/index.php#prettyPhoto. Timetable.

[5] https://horario-estudio.doodle.com/create/options. Doodle.

[6] https://empresearandom.woffu.com/#/requests. Woffu.

[7] https://app.smartsheet.com/sheets/5mRxHXj8rfgp3gFCwcxwWjMP2GgwX6gX6P3PwPQ1?view=grid. smartsheet.

[8] https://geaportal.ucm.es/. geaportal.

[9] https://kanbantool.com/es/metodologia-kanban. Kanban.

[10] https://www.php.net/manual/es/intro-whatcando.php. Php.

[11] https://github.com/jquery/jquery. jquery.

[12] https://github.com/jquery/jquery-ui. jquery-ui.

[13] https://www.w3schools.com/css/. Css.

[14] https://github.com/twbs/bootstrap. Bootstrap.

[15] https://trello.com/. Trello.

[16] https://www.overleaf.com. overleaf.

[17] https://www.ucm.es/data/cont/media/www/pag-114936/E.10.pdf. Matriculaciones 17-18.

[18] http://phpsysinfo.github.io/phpsysinfo/. phpsysinfo.

[19] https://aws.amazon.com/es/. Amazon web services.

[20] https://aws.amazon.com/es/ec2/. Amazon ec2.

[21] https://aws.amazon.com/es/ebs/. Amazon ebs.

[22] https://aws.amazon.com/es/s3/. Amazon s3.

[23] https://www.adobe.com/es/products/xd.html. Adobe xd.

[24] https://www.ucm.es/oficina-de-software-libre. Otea.

[25] https://www.ucm.es/oficina-de-software-libre/catalogo-alejandria. Catálogo alejandría.

Javier García Lorenzo
Carlos Lozano Casado
MAYO 2019
Ult. actualización 22 de mayo de 2019